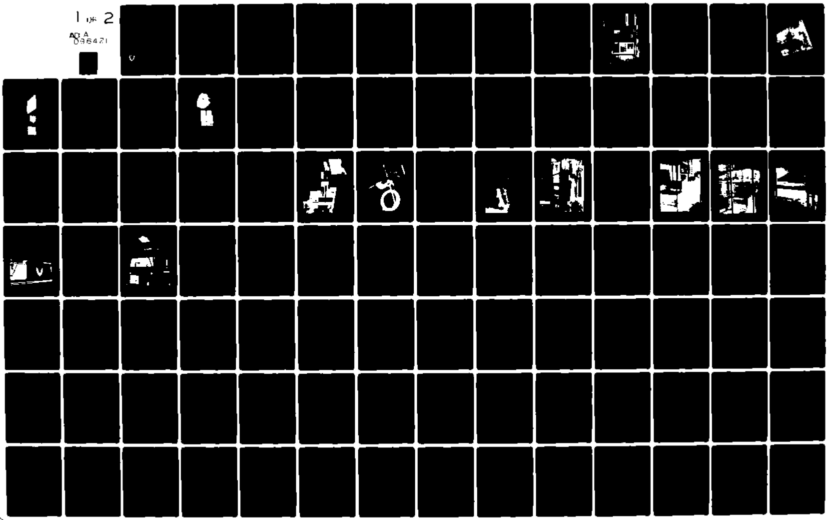
AD-A096 421 ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND ST LO--ETC F/G 17/7
THE DEVELOPMENT AND TESTING OF THE NAVSTAR GLOBAL POSITIONING S--ETC(U)
FEB 81 J GRAY
UNCLASSIFIED USAAVRADCOM-TR-80-E-3 NL

1 of 2
AD A 096421



ADA096421

AVRADCOM

Technical Report -80-E-3

THE DEVELOPMENT AND TESTING OF THE NAVSTAR GLOBAL POSITIONING SYSTEM/DOPPLER RADAR VELOCITY SENSOR HEADING REFERENCE SYSTEM (GDHED).

JACK GRAY

FEBRUARY 1981

Research and Development Technical Report
Aviation Research and Development Command

NOTICES

Disclaimers

Disposition

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
AVRADCOM TR 80-E-3

2. GOVT ACCESSION NO.
AD-A096 421

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
The Development and Testing of the NAVSTAR Global Positioning System/Doppler Radar Velocity Sensor Heading Reference System (GDHED)

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Jack Gray

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Headquarters
US Army Avionics R&D Activity
ATTN: DAVAA-N
Fort Monmouth, NJ 07703

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
61110191A001101YA50Z-0

11. CONTROLLING OFFICE NAME AND ADDRESS
Headquarters
US Army Avionics R&D Activity
ATTN: DAVAA-N
Fort Monmouth, NJ 07703

12. REPORT DATE
February 1981

13. NUMBER OF PAGES
145

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

N

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Global Positioning System
Satellite Navigation
Doppler Navigation
Heading Reference Systems
Hybrid Navigation Systems
Navigation Computers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A GPS/Doppler Heading Reference system was developed and field tested in a van for evaluation of performance in the Fort Monmouth, NJ area. Earth-referenced velocities derived from the GPS set and body referenced velocities derived from the AN/ASN-128 Doppler Set were combined in a separate HYBRID computer to determine true vehicle heading. The results of these tests successfully demonstrated that a new method of providing non-magnetic, non-gyrocompass vehicle heading is possible.

CONTENTS

APPENDICES

TABLES

A 23

Page

FIGURES

1. INTRODUCTION

The GPS/Doppler Heading Reference (GDHED) program is an Independent Laboratory, In-House Research (ILIR) funded program. It is an exploratory development of a navigation concept whereby an accurate heading reference for airborne and ground vehicles is generated by combining Earth Referenced Velocities derived from the Navigation Satellite Timing and Ranging Global Positioning System (NAVSTAR GPS) with Body Referenced velocities derived by the AN/ASN-128 Doppler Radar Velocity Sensor (DRVS).

2. OBJECTIVE

The objective is to demonstrate the Heading Accuracy of a GPS/Doppler Heading Reference System.

3. BACKGROUND

The concept of a GPS/Doppler Heading Reference System was disclosed during the first quarter of FY-80 in the form of a patent application.[1] The purpose of the invention was to provide a non-magnetic, non-gyrocompass true Heading Reference System for Army airborne and ground vehicular users. Probable uses of this invention would be to improve compasses, via Kalman filtering, in areas of the Earth where, due to variations of the Earth's magnetic field, magnetic compasses may not be used, and with gyro compass-based systems to reduce gyrocompass errors through dynamic calibration/alignment. Also, where primary heading references may be a casualty, the proposed approach would yield a "fall-back" heading reference.

4. *SYSTEM TECHNICAL APPROACH*

The design for a GDHED system required the integration of five separate systems. These systems included:

GPS User Equipment

AN/ASN-128 Doppler Radar Sensor

Attitude Reference System

Computer System

Auxiliary System

The systems approach involved in this development required the successful execution of the following tasks:

a. Acquisition of Prime Equipments.

(1) Global Positioning System. The GPS equipment used was the Texas Instruments' "High Dynamic User Equipment" Advanced Development Model[2] (Figure 1). A functional block diagram of the GPS subsystem is depicted in Figure 2.

[1] "Global Positioning System/Doppler Radar Hybrid Velocity Derived Heading Reference System," Patent Docket No. D-2071, Jack Gray, Inventor.
[2] "Texas Instruments Phase 1 GPS User Equipment," M. J. Borel, et. al., NAVIGATION: Journal of the Institute of Navigation, Vol. 25, No. 2, Summer 1978.

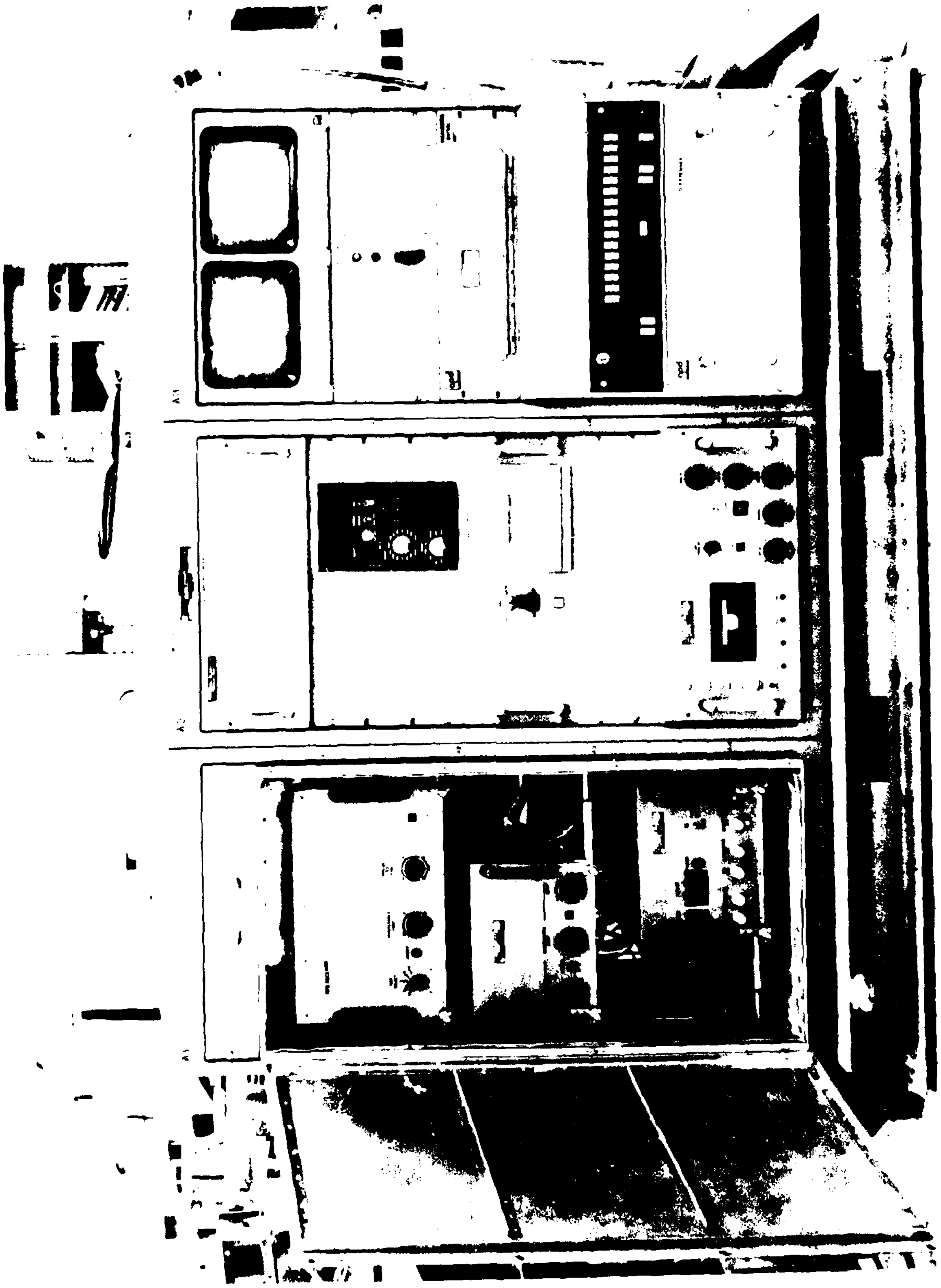


Figure 1. Global positioning system user equipment

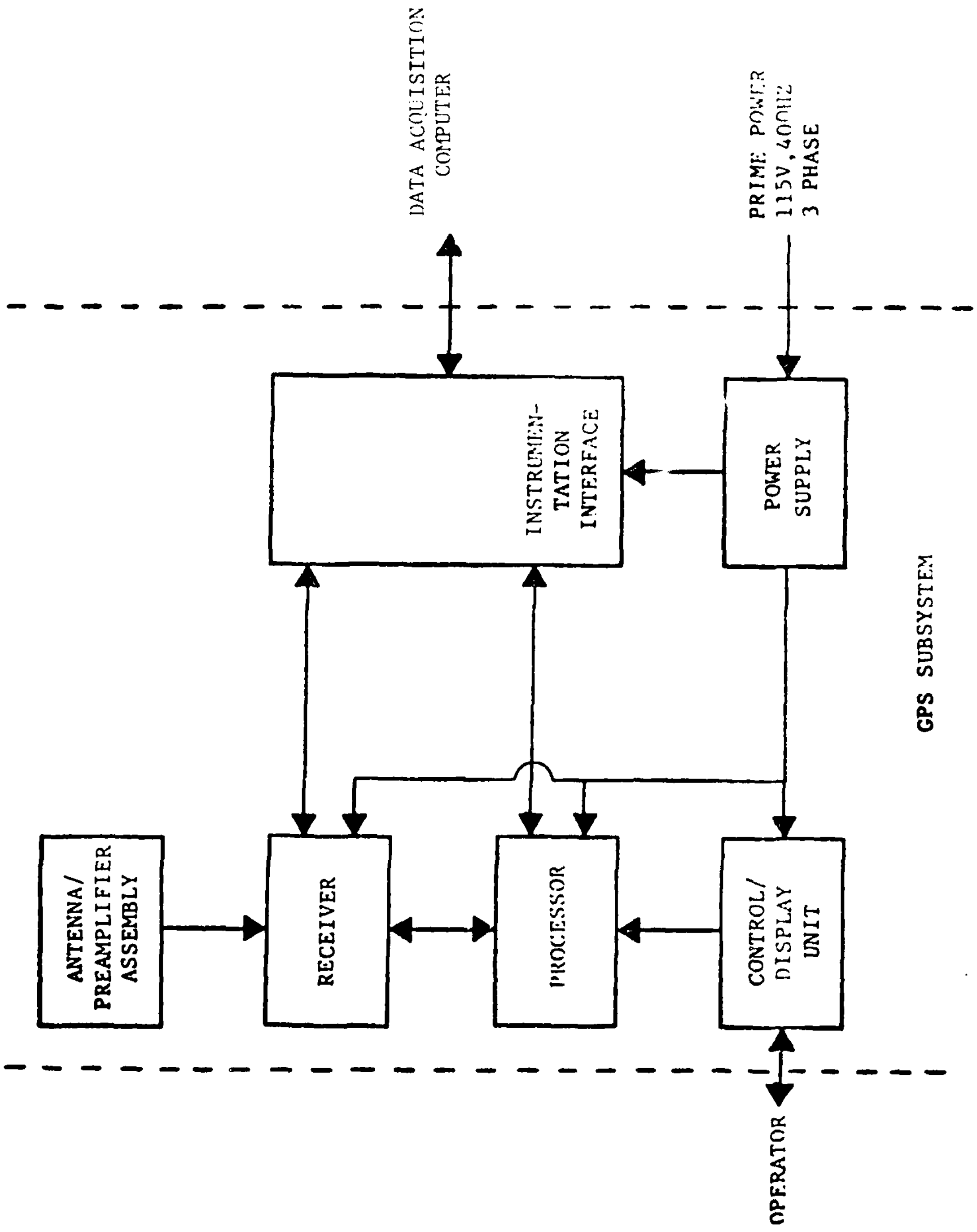


Figure 2. GPS ubsystem function flow diagram

The GPS antenna used was a dual frequency (1227.6 MHz, 1575.4 MHz), 20 MHz bandwidth, right-hand circularly polarized, volute antenna designed by CHU Associates, CA, whose antenna VSWR amd radiation patterns are depicted in Appendix A.

The GPS preamplifier used was Magnavox's GPS Phase 1 "X-Set" Preamplifier whose characteristics are presented in Appendix B.

The GPS subsystem physically consists of the following units:

UNIT	FUNCTION
Antenna/Preamplifier Assembly	Receives RF signals from up to five satellites and filters, amplifies and transmits the signals to the receiver/processor assembly (see Figure 3).
Receiver	Consists of five single channel receivers connected to a matrix switch output, and a check module for system timing. Acquires, tracks, demodulates, and performs necessary processing to derive pseudo range, pseudo range rate, down-link data and system time from the satellite signals.
Navigation Processor	Provides overall GPS subsystem control and performs navigation calculations.
Instrumentation Interface Unit (IIU)	Provides intercommunications between the receiver/processor and data acquisition computer. This unit also loads the navigation programs into processor memory.
Control/Display Unit	Provides the human interface and operating mode control functions for overall receiver operation. The unit consists of a multifunction keyboard for receiver mode and navigation display control and alpha numeric displays for monitoring of navigation parameters.

(2) AN/ASN-128 Doppler Radar Velocity Sensor. The AN/ASN-128 Doppler Radar Velocity Sensor used was the Singer Kearfott "Lightweight Doppler Navigation System" Engineering Development Model[3] (Figure 4). A functional block diagram of the Doppler subsystem is depicted in Figure 5. The Doppler Subsystem physically consists of the following three units:

[3] Lightweight Doppler Navigation System (LDNS), ETO-1201A, The Singer Company, Kearfott Division, 16 March 1976.

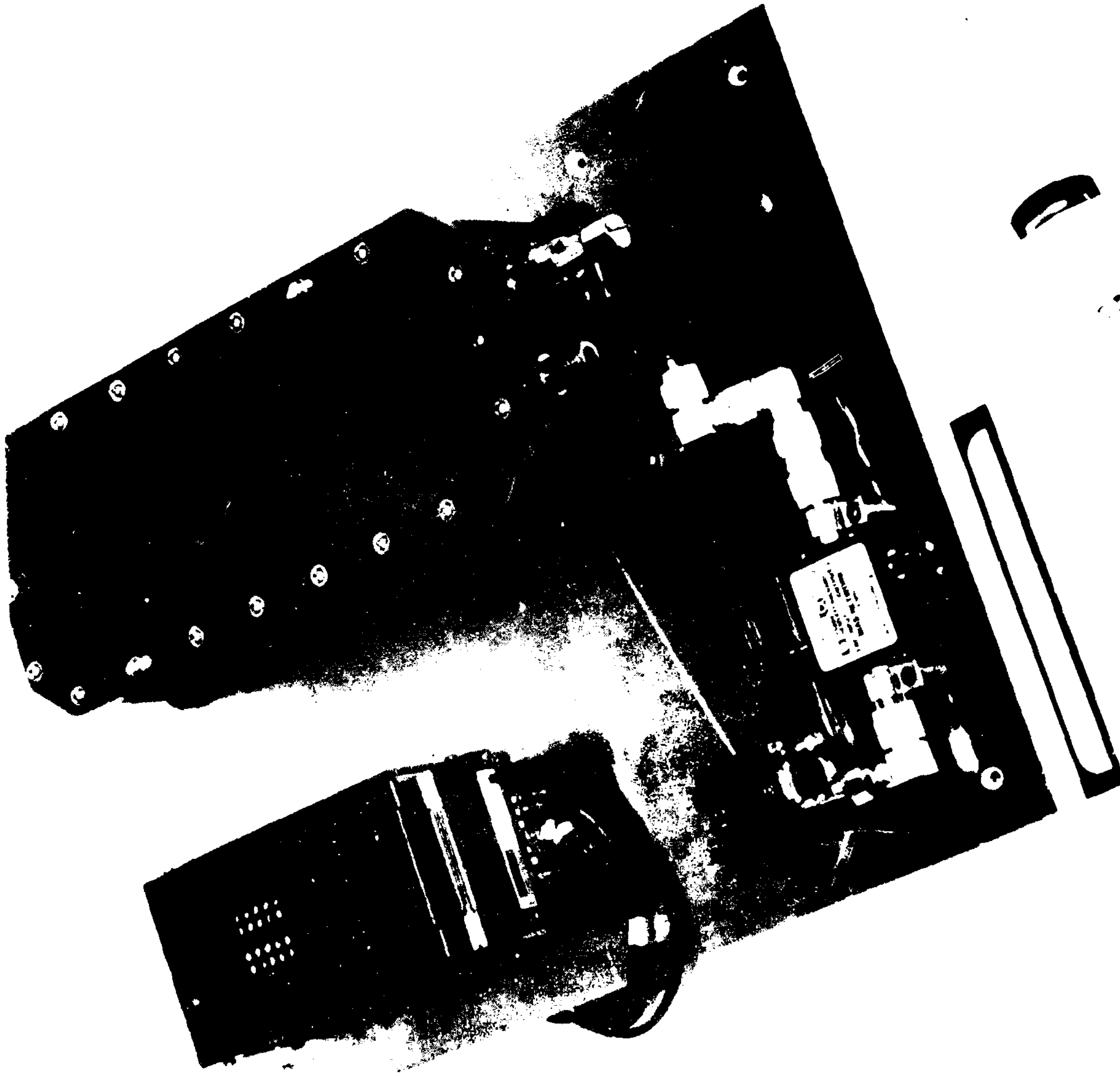


Figure 3. [illegible]

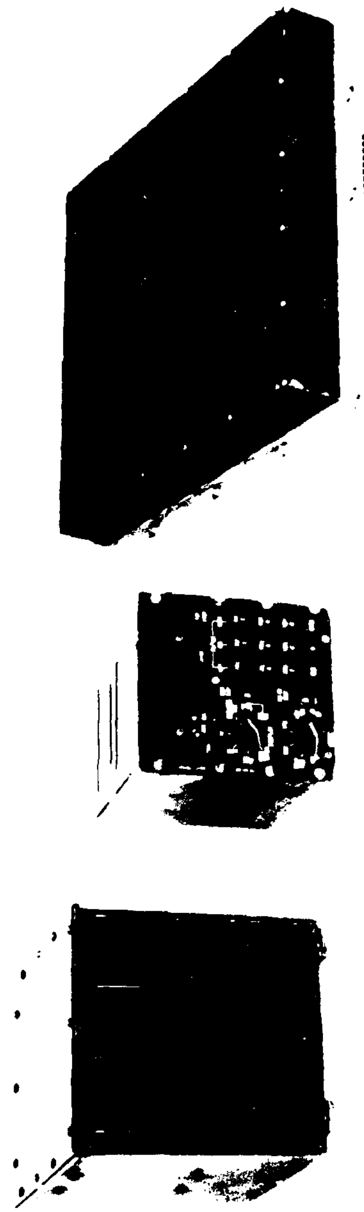


Figure 4. AN/ASN-128 Doppler equipment

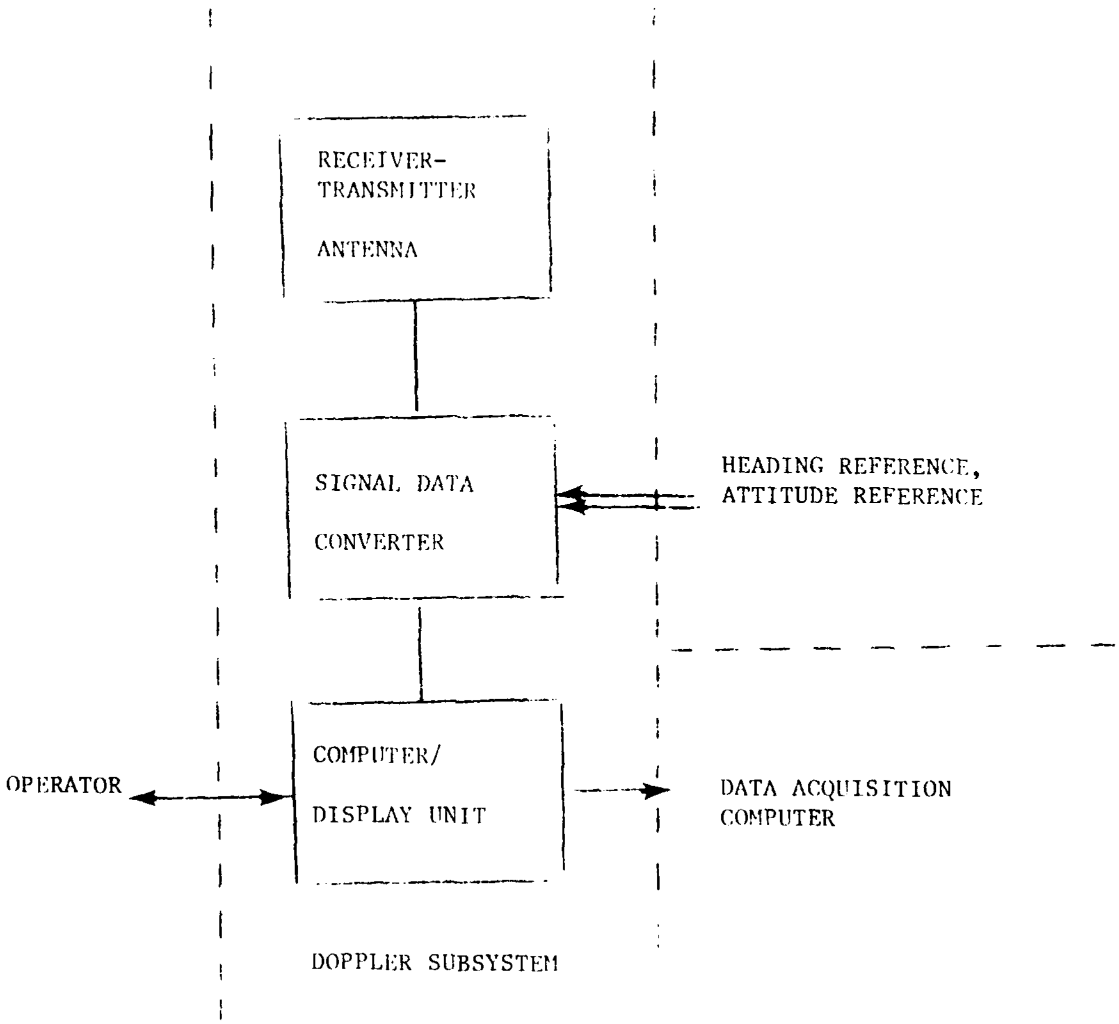


Figure 5. Doppler subsystem function flow diagram

UNIT	FUNCTION
Receiver-Transmitter Antenna (RTA)	Transmits RF(13.325 GHz) energy toward the ground in four non-coplanar beams/ measures the four Doppler frequency shifts in the backscattered energy/ transmits the four Doppler frequency shifts (in terms of components along the beam directions) to the SDC.
Signal Data Converter (SDC)	Accepts heading and vertical reference synchro signals, and along with Doppler beam velocities transmits serial digital outputs to the CDU computer.
Computer Display Unit (CDU)	Accepts from the SDC beam velocities, Heading, Roll, and Pitch. Performs the Navigation Computations/provides intercommunication between the DRVS and Data Acquisition Equipment.

(3) Attitude reference system. An Air Force Model MD-1 Displacement gyroscope[4] was the Attitude Reference system used. It is a 2 degree of freedom gyro which generates pitch and roll signals with a nominal error of 3-degrees (standard deviation). These signals are in the form of three wire synchro signals.

(4) Hybrid computer system. The computer selected for the GDHED field test was the AN/UYK-34 MIL-SPEC computer manufactured by ROLM Corporation as their Model 1650 (Figure 6). This is a high speed, 16-bit word general purpose computer. It has four general purpose accumulator registers and contains 32-K core memory. It is modular and has standard interfaces for a teleprinter/magnetic casette tape recorder, and paper tape reader/punch. Driven by a real-time clock, it includes a hardware floating point capability. It is also expandable, providing slots for special interfaces, such as the Doppler ARINC serial bus interface.

The controlling element of the HYBRID computer is the Computer Control Panel (CCP). The CCP is a ruggedized portable unit which has the capability of the following major functions:

(a) it provides direct control of the HYBRID computer. This includes the capability to examine and load memory so as to modify programs online

(b) it provides the programmer the capability to monitor and debug the software either in real-time or off-line.

(c) it provides memory load and verify capability by use of a separate paper tape reader.

[4]MIL-SPEC MIL-G-25597D (USAF), Gyroscope, Displacement, Roll and Pitch, Type MD-1, 6 August 1973.

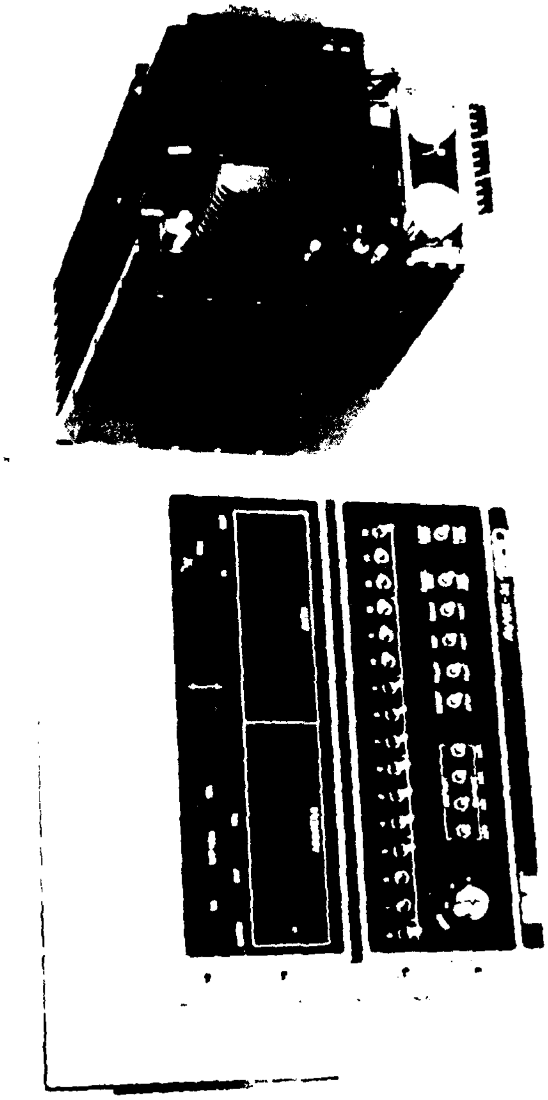


Figure 6. CPS [illegible] hybrid computer

(d) it provides diagnostic capability to separate hardware and software faults and to troubleshoot hardware faults in the computer.

The CCP is implemented as part of an I/O chassis with the logic cards mounted in the chassis behind the panel. It is directly connected to the main Data, Address, and Control bus in the computer so that direct access is provided to all modules.

The CCP gives the operator complete control over computer operation. From the panel, the operator may select various modes of execution, load and verify the memory, readout contents of memory, execute instructions from front panel switches, and display register contents and status.

(5) Auxiliary system. The Auxiliary system consists of the standard computer peripheral equipments that are used to load and print out computer data. These are listed below, along with a brief description.

UNIT	FUNCTION
Teletype Unit (T.I. 733)	Real time display/hardcopy of GDHED system performance.
Paper Tape Reader/Punch (IOMEC)	Loads/punches out GDHED software into/out of the HYBRID computer.
Magnetic Casette Tape Unit (T.I. 733)	GDHED data logging for post-mission analysis.

b. Development of Hybrid Computer Real Time Stand Alone (RTOS) Software. The initial step in generating the software was the invention of the GDHED concept in the patent application. Having established this requirement, which mathematically formulated the equations which were ultimately implemented, an error analysis was performed. This analysis identified all error sources and determined their contribution to the total system error. The validity of the mathematical formulations was performed on a general purpose large scale computer along with the error analysis (see Appendix C for program listing).

(1) GDHED error analysis. The potential performance to be expected of the GDHED system is fairly easy to deduce from an analytical approach. Expressions for the North (V_N), East (V_E), Along-Track (V_H) and Cross-Track (V_D) velocities of the vehicle are first obtained, and then the GDHED equation is derived. The GDHED equation is then examined for sensitivity to errors in the variables. Following this, the sensitivity analysis is extended to a statistical consideration of the overall error which is likely to result.

(2) Derivation of GHDED equations. Before an analysis of the GDHED error can be performed, a brief description of the two velocity measuring systems is in order including the equations defining the problem geometry. In the derivation, the heading is described as a function of those parameters which are both pertinent to the problem geometry and capable of being measured.

(a) NAVSTAR GPS. The Global Positioning System is a satellite referenced radio navigation system. It basically consists of a constellation of satellites and a ground tracking network (Figure 7). The ground tracking network periodically measures and updates the ephemeride of each satellite and keeps all the satellite clocks synchronized. The satellites continuously transmit orthgonally binary coded ranging signals to the user. These coded signals also serve to identify which satellite signal is being received. By using a code correlation detector, the GPS equipment can measure the time delay of the transmitted signal. This time delay measurement not only includes the signal propagation delay but also the clock bias and clock bias rate differences between the user equipment and the satellites.

A GPS system user equipped with passive one-way ranging equipment (antenna, receiver, and computer) can determine his position and velocity by measuring the GPS signal time of arrival together with the GPS signal doppler. The antenna receives all available satellite signals and the receiver selects four of the satellites to establish four independent pseudo-ranges and pseudo-range rates. The user computer then solves the navigation equations to derive the user position/velocity using those satellites.

The GPS outputs used by the GDHED system are the North velocity ($V_{N(GPS)}$) and East velocity ($V_{E(GPS)}$).

(b) AN/ASN-128 Doppler radar sensor system. The AN/ASN-128 Doppler Radar System measures velocities by sensing the reflected Doppler shifts of a downward transmitted microwave signal along four shaped antenna beams (Figure 7). The Doppler frequency tracker determines the center of power (mean frequency) of the noiselike Doppler frequency spectrum obtained from the ground echo and through mixing the Doppler signal with a tracking oscillator derives the Doppler shift for each of the four beams. Since the Doppler antenna is attached to the vehicle, the antenna beam geometry is defined in terms of vehicle coordinates. The measured Doppler frequencies are processed by the Doppler computer, and converted to vehicle coordinate velocities.

The Doppler generated vehicle body axis velocities V_X, V_Y, V_Z must be tempered due to pitch and roll motions of the vehicle since the derived Doppler coordinate velocities apply only when the vehicle is moving straight and level (i.e., when vehicle body axis V_X, V_Y, V_Z coincide with the V_H, V_D, V_V axis).

If the vehicle experiences a change in Pitch (P) (i.e., a rotation about the V_D axis (Figure 8) the coordinate transformation relationship is

$$\begin{bmatrix} V_H \\ V_D \\ -V_V \end{bmatrix} = \begin{bmatrix} \cos(P) & 0 & \sin(P) \\ 0 & 1 & 0 \\ -\sin(P) & 0 & \cos(P) \end{bmatrix} \cdot \begin{bmatrix} V_X^1 \\ V_Y^1 \\ V_Z^1 \end{bmatrix}$$

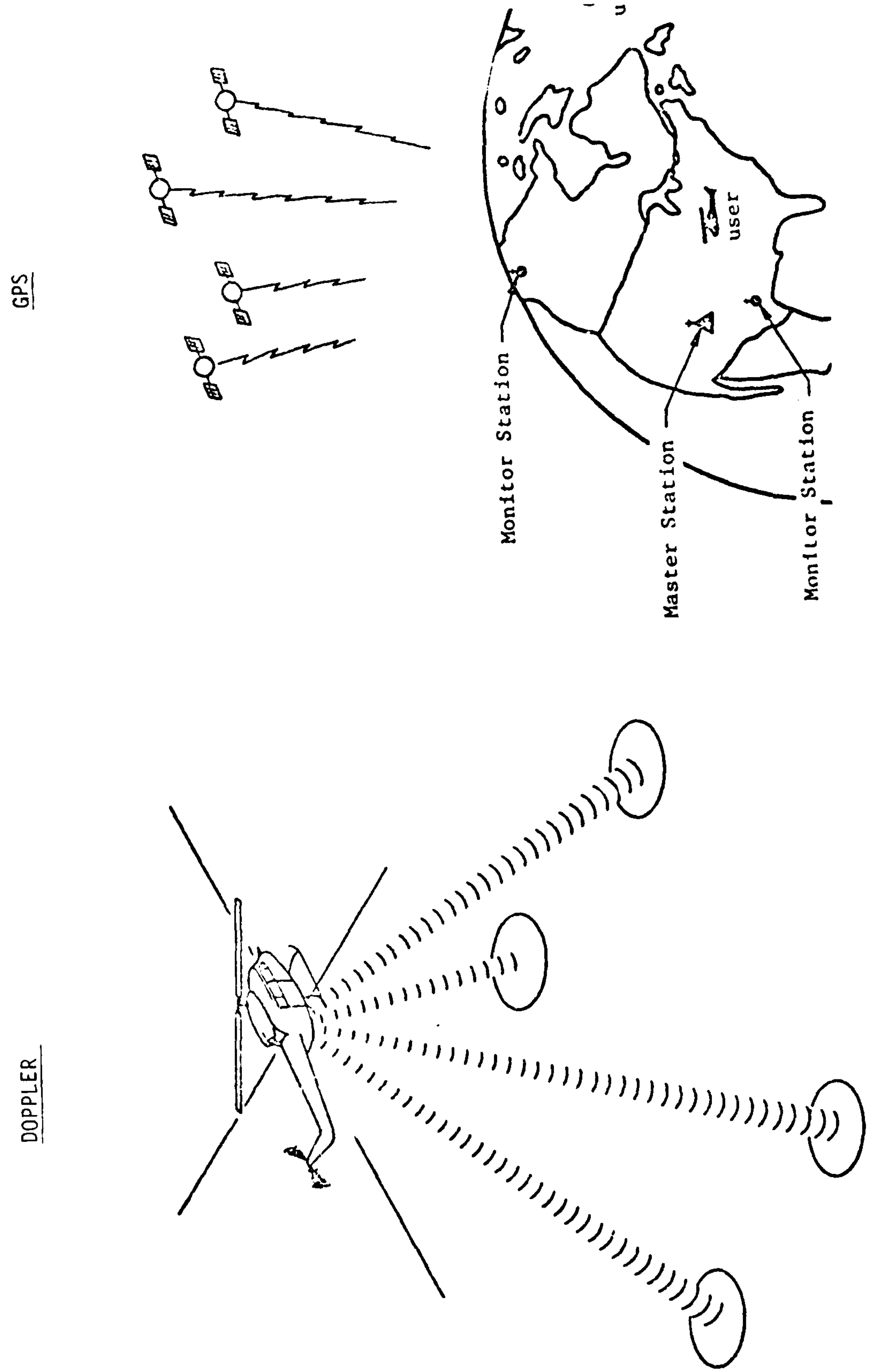


Figure 7. GPS and Doppler stand alone navigation systems

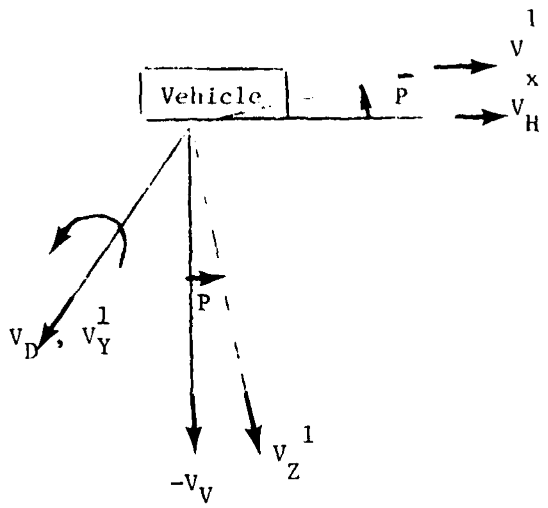


Figure 8. Doppler coordinate system for pitch change

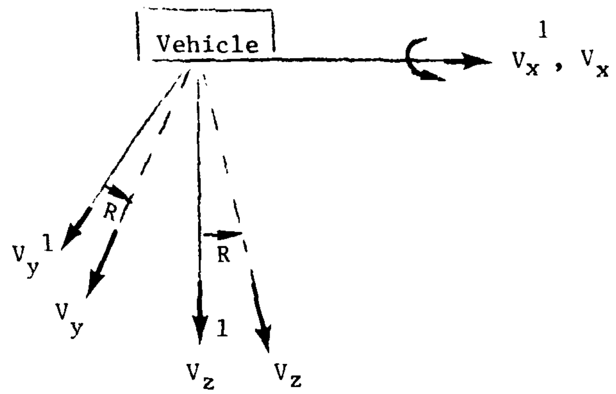


Figure 9. Doppler coordinate system for roll change

Referring to Figure 9, a rotation about the V_X axis results in a change in vehicle Roll (R) and is described by the following matrix,

$$\begin{bmatrix} V_X^1 \\ V_Y^1 \\ V_Z^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(R), & -\sin(R) \\ 0 & \sin(R), & \cos(R) \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \qquad (2)$$

Multiplying both matrices for pitch and roll changes,

$$\begin{bmatrix} V_H \\ V_D \\ -V_V \end{bmatrix} = \begin{bmatrix} \cos(P), & \sin(P)\ \sin(R), & \sin(P)\ \cos(R) \\ 0\ , & \cos(R), & -\sin(R) \\ -\sin(P), & \cos(P)\ \sin(R), & \cos(P)\ \cos(R) \end{bmatrix} \cdot \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \qquad (3)$$

When a Vertical Gyro is mounted along the heading and drift axis of the vehicle, the roll and pitch values will be provided to the Doppler, and the desired heading velocity (V_H) and Drift Velocity (V_D) can be calculated, since V_X, V_Y, and V_Z had already been determined.

The V_H and V_D axis, being the straight and level flight axes of the vehicle, may be viewed on a Spherical Earth coordinate system as a tangent plane perpendicular to an imaginary radial (R_e) emanating from the center of the Earth (Figure 10). These Doppler velocities, V_H and V_D are the remaining velocities required in the GDHED system.

(c) GPS/Doppler heading reference system description. The coordinate system shown in Figure 11 will be used to obtain the GPS/Doppler Radar Sensor velocity derived heading equations. These equations are as follows:

$$V_N = V_H \cos(H) - V_D \sin(H) \qquad (4)$$

$$V_E = V_D \cos(H) + V_H \sin(H) \qquad (5)$$

Solving for cos(H) and sin(H)

$$\cos(H) = \frac{V_N V_H + V_D V_E}{V_H^2 + V_D^2}, \quad \sin(H) = \frac{V_H V_E - V_N V_D}{V_H^2 + V_D^2}$$

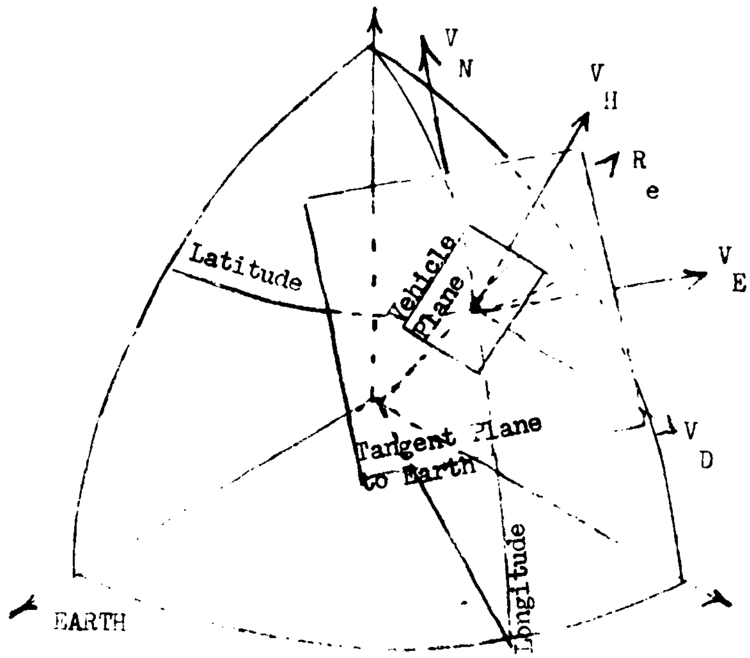


Figure 10. Doppler velocities V_H, V_D geometry

V_N = North velocity

V_E = East velocity

V_H = Heading velocity

V_D = Drift velocity

H = Heading

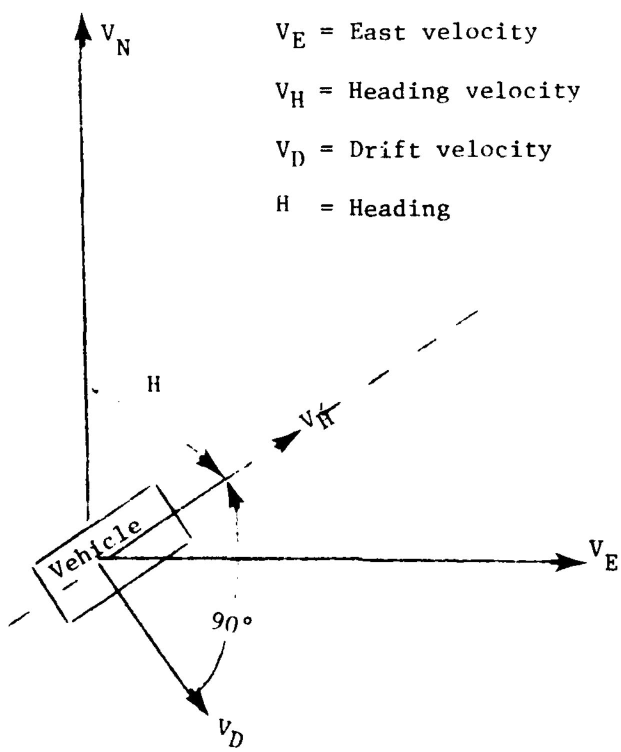


Figure 11. GPS/Doppler heading reference geometry

therefore,

$$\text{Heading} = H = \text{Arctan}\ \frac{V_H V_E - V_N V_D}{V_N V_H + V_D V_E} \qquad (6)$$

The V_N, V_E velocities would be provided by $V_{N(GPS)}$, $V_{E(GPS)}$. The V_H, V_D velocities would be provided by the Doppler Radar Sensor.

(2) Derivation of GDHED error equations. A linear error analysis was performed on the GDHED variables; namely, Pitch (P), Roll (R), North Velocity (V_N), East Velocity (V_E), and Body-fixed coordinate velocities (V_x, V_y, V_z)

Although,

$$H = H\ (V_H, V_E, V_D, V_N),$$

V_D and V_H are dependent upon other variables, namely,

$$V_H = V_H(V_x, V_y, V_z, P, R)$$

and $$V_D = V_D(V_x, V_y, V_z, P, R)$$

therefore, $H = H(V_N, V_E, V_x, V_y, V_z, P, R)$

Letting

$$\xi_1 = P$$

$$\xi_2 = R$$

$$\xi_3 = V_x$$

$$\xi_4 = V_y$$

$$\xi_5 = V_z$$

$$\xi_6 = V_N$$

$$\xi_7 = V_E$$

and taking the total differential of H yields,

$$dH = \sum_{i=1}^{7} \frac{\partial H}{\partial \xi_i}\, d\xi_i$$

letting $E_H = d_H$, where E_H is the error in the computed value of Heading, and letting $de_i = \xi_i$, where e_i is the error in the measurement of the ith variable,

Then

$$E_H = \sum_{i=1}^{7} \frac{\partial H}{\partial \xi_i} e_i \tag{7}$$

Calculating the partial derivative of H with respect to the ξ_i:

$$H^1 = \frac{V_H V_E - V_N V_D}{V_N V_H + V_D V_E} \tag{8}$$

$$\frac{\partial H}{\partial V_H} = \frac{1}{1+(H^1)^2} \frac{\partial H^1}{\partial V_H} \tag{9}$$

$$\frac{\partial H^1}{\partial V_H} = \frac{V_D (V_E^2+V_N^2)}{(V_N V_H + V_D V_E)^2} \tag{10}$$

$$\frac{\partial H^1}{\partial V_E} = \frac{V_N (V_H^2+V_D^2)}{(V_N V_H+V_D V_E)^2} \tag{11}$$

$$\frac{\partial H}{\partial V_E} = \frac{1}{1+(H^1)^2} \frac{\partial H^1}{\partial V_E} \tag{12}$$

$$\frac{\partial H^1}{\partial V_D} = \frac{-V_H (V_E^2+V_N^2)}{(V_N V_H+V_D V_E)^2} \tag{13}$$

$$\frac{\partial H}{\partial V_D} = \frac{1}{1+(H^1)^2} \frac{\partial H^1}{\partial V_D} \tag{14}$$

$$\frac{\partial H^1}{\partial V_N} = \frac{-V_E (V_H^2+V_D^2)}{(V_N V_H+V_D V_E)^2} \tag{15}$$

$$\frac{\partial H}{\partial V_N} = \frac{1}{1+(H^1)^2} \frac{\partial H^1}{\partial V_N} \tag{16}$$

$$\frac{\partial V_H}{\partial P} = -V_x \text{ SIN } (P) + V_y \text{ COS}(P) \text{ SIN}(R) + V_z \text{ COS}(P) \text{ COS}(R) \tag{17}$$

$$\frac{\partial V_H}{\partial R} = V_y \; COS(R)SIN(P) \; -V_z SIN(R)SIN(P) \qquad (18)$$

$$\frac{\partial V_H}{\partial V_x} = COS(P) \qquad (19)$$

$$\frac{\partial V_H}{\partial V_y} = SIN(P) \; SIN(R) \qquad (20)$$

$$\frac{\partial V_H}{\partial V_z} = SIN(P) \; COS(R) \qquad (21)$$

$$\frac{\partial V_D}{\partial P} = \phi \qquad (22)$$

$$\frac{\partial V_D}{\partial R} = -V_y \; SIN(R) \; -V_z \; COS(R) \qquad (23)$$

$$\frac{\partial V_D}{\partial V_x} = \phi \qquad (24)$$

$$\frac{\partial V_D}{\partial V_y} = -SIN(R) \qquad (25)$$

$$\frac{\partial V_D}{\partial V_z} = -V_z COS(R) \qquad (26)$$

Combining equations (8) through (26) into equation (7) we obtain

$$E_H = \frac{\partial H}{\partial V_H} dV_H + \frac{\partial H}{\partial V_E} dV_E + \frac{\partial H}{\partial V_D} dV_D + \frac{\partial H}{\partial V_N} dV_N \qquad (27)$$

where

$$dV_H = \frac{\partial V_H}{\partial P} + dP + \frac{\partial V_H}{\partial R} dR + \frac{\partial V_H}{\partial V_x} dV_x + \frac{\partial V_H}{\partial V_y} dV_y + \frac{\partial V_H}{\partial V_z} dV_z \qquad (28)$$

$$dV_D = \frac{\partial V_D}{\partial P} dP + \frac{\partial V_D}{\partial R} dR + \frac{\partial V_D}{\partial V_x} dV_x + \frac{\partial V_D}{\partial V_y} dV_y + \frac{\partial V_D}{\partial V_z} dV_z \qquad (29)$$

Equation (27) defines the total error in heading as a function of the seven GDHED variables V_N, V_E, V_x, V_y, V_z, P, R and the seven associated errors, ϵ_i.

(4) Statistical simulation results. In order to determine the total GDHED statistical error, the following conditions were assumed:

If we assume that the seven GDHED variables have been assigned specific values, then the variable elements in equation (27) are fixed. Let these elements be denoted C_i. Next assume that the e_i are independent, normally distributed random variables with density functions.

$$f_i(\eta) = N\,[\mu_i,\ \sigma_i] = \frac{1}{\sigma_i\sqrt{2\pi}}\ e^{-(\eta-\mu_i)^2/2\sigma_i^2}$$

It can then be shown that the density function corresponding to the random variable

$$E_H = \sum_{i=1}^{7} C_i\ e_i$$

is normal with mean

$$\bar{\mu}_H = \sum_{i=1}^{7} C_i\ \mu_i$$

and standard deviation

$$\bar{\sigma}_H = \left(\sum_{i=1}^{7} (C_i\ \sigma_i)^2\right)^{1/2}$$

Based upon these asumptions, several techniques were used in performing the simulation. The first consisted of choosing nominal values of V_T*, P, H, and R and allowing them to vary in increments over predetermined ranges.

Tables 1 and 2 list the standard deviations corresponding to each e_i and the maximum, minimum, and incremented values of the GDHED parameters.

$*V_T = (V_x^2 + V_y^2 + V_z^2)^{1/2}$

TABLE 1. STANDARD DEVIATIONS AND MEANS CORRESPONDING TO EACH VARIABLE, e_i [5,6]

ERROR	STANDARD DEVIATION (KNOTS)	MEAN
e_{V_N}	0.1	0
e_{V_E}	0.1	0
e_{V_X}	$0.0025\ V_T + 0.1$	0
e_{V_y}	$0.0025\ V_T + 0.1$	0
e_{V_z}	$0.001\ V_T + 0.05$	0
e_P	3.0°	0
e_R	3.0°	0

TABLE 2. MAXIMUM, MINIMUM, AND INCREMENTAL VALUES OF THE GDHED PARAMETERS

PARAMETER	MINIMUM	MAXIMUM	INCREMENT
V_T	10 knots	100 knots	10 knots
H	0°	355°	5°
P	1°	46°	5°
R	1°	31°	5°

[5] Lightweight Doppler Navigation System (LDNS) Electronics Command Development Specification, EL-SS-1050-001A. [illegible] June 1973.

[6] Journal of the Institute of Navigation, Summer 1978 - Vol. 25, #2, "Principples of Operation of NAVSTAR" - R. J. Milliken and C. J. Zoller, p. 95-106.

Allowing these parameters to assume, progressively, their respective incremental values, 21,600 GDHED states may be obtained.

The resultant mean and standard deviation of the simulated GDHED system which was run under the constraints listed in Tables 2 and 3 were:

GDHED mean error = 0.04°

GDHED standard deviation = 0.37°

Realizing that the Doppler velocity errors were dependent upon total vehicle velocity, V_T, it became of interest to investigate the expected total GDHED error whereby all parameters and errors listed in Tables 1 and 2 were left unaltered except that the parameter V_T was held constant throughout each run. The results of each run were then a function of V_T, which assumed values ranging from 5 knots for the first run, to 100 knots for the tenth run. Table 3 shows the results of these runs.

TABLE 3. CONSTANT VELOCITY STATES AND TOTAL GDHED ERROR

V_T (KNOTS)	GDHED ERROR MEAN	GDHED ERROR STD DEVIATION
5	0.04°	0.24°
10	0.03°	0.21°
20	0.02°	0.22°
30	0.01°	0.25°
40	0.01°	0.28°
50	0.00°	0.31°
60	0.00°	0.35°
70	0.01°	0.39°
80	0.02°	0.44°
90	0.03°	0.49°
100	0.04°	0.54°

By examining Table 3 we see a slight improvement of the GDHED system if the vehicle velocity is held constant, and low, as compared to its performance over a wide range of variable velocities.

Another series of runs were made to determine the effects of degraded GPS performance upon the GDHED system. In this scenario the variables and errors depicted in Tables 1 and 2 remained the same, with the exception of the GPS velocity errors, e_{V_N} and e_{V_E}. These GPS velocity errors were varied for each run, starting from minimum of 0.1 knots for the first run, to a maximum of 5.0 knots for the fifth run. Table 4 shows the results of these runs.

TABLE 4. GPS ERROR STATES AND TOTAL GDHED ERROR

GPS ERROR STATE $e_{V_N} = e_{V_E}$ (KNOTS)	GDHED SYSTEM ERROR MEAN	GDHED SYSTEM ERROR STANDARD DEVIATION
0.1	0.04°	0.37°
0.5	0.04°	0.40°
1.0	0.04°	0.50°
2.0	0.04°	0.91°
5.0	0.03°	3.80°

It should be pointed out that all the runs assumed that $V_x = V_y = V_z$ throughout. This condition therefore simulates vehicle is not only moving horizontally, but also vertically.

(5) GDHED RTOS software. Having verified the GDHED mathematical formulations, explicit mechanization equations had to be generated.

The GDHED software was designed to meet future applications of an integrated GPS-Doppler system as well as the immediate needs. It is completely modular in concept according to function. Each function consists of driver, processor, and monitor subfunctions. The driver function provides the logic necessary to effect communication and data transfer between computer and an external device. The processor subfunction performs the required data manipulation and/or computations. The monitor subfunction provides the interface between a module and the Real-Time Executive (RTE).

Written in Assembly language, and entirely core-resident, the GDHED software is partioned as follows:

(a) The Real Time Executive module controls the internal job flow. These jobs include generalized priority scheduling, input/output (I/O) and interrupt management and error management.

(b) The Initialization module performs the initialization function on system startup. It initializes the status of each program module and activates the modules necessary to begin system operation.

(c) The GPS Interrupt module receives navigation/system status data from the GPS set, monitors system alerts of changes in communication status, and performs reasonableness checks on incoming data.

(d) The Doppler Interrupt module performs identically as the GPS Interrupt module for navigation/system status data from the Doppler set.

(e) The Auxiliary System Interrupt module enables the keyboard for the operator; enables system status and data printouts, monitors teletype status in order to insure orderly transfer of input and output data; decodes keyboard inputs; formats data for printout loading.

(f) The Math Module provides double precision floating point trigonometric functions for the GDHED calculations.

(g) The Formatter module decodes/packs, floating point/fixed point, input/output, data/messages into ASCII input/output buffers.

(h) The GDHED module contains the mathematical algorithms of the system and outputs the results to the auxiliary equipment.

Figure 12 depicts the software modules used in the GDHED system. These seven modules provide a modifiable, expandable, and flexible navigation software system. Appendix D provides a complete listing of the GDHED RTOS programs. These programs were developed in this Activity's Hybrid Computer Laboratory (Figure 13) using ROLM's Real-Time Disk Operating System (RDOS).

After the programs were successfully tested under RDOS, all the programs were transferred onto paper tape. This GDHED RTOS tape when loaded via the paper tape reader into the HYBRID computer enables a real-time, stand-alone, non-RDOS, GDHED mode of operation.

c. Design of GPS and Doppler Interface Units. The GPS and Doppler Interface Units were designed in-house. The GPS Interface unit (Figure 14) was designed to provide a 16-bit parallel, 256-word data block transfer via Direct Memory Access (DMA) to the HYBRID computer. Buffers, time delay/synchronization circuits were needed since the GPS set employs dual differential logic while the HYBRID computer operated with TTL gates; and transfer rates had to be adjusted between the two systems. The DMA operation was designed such that when enabled it came under the control of the GPS set. Therefore, only when the GPS set was ready to send out data could the HYBRID computer process it.

The Doppler Interface Unit was designed to provide for the transfer of the Doppler Auxiliary digital output signals as specified.[7] The Doppler set serially transmits both 32-bit Binary and BCD coded data. The Doppler interface converted the 32-bit serial data into 16-bit parallel slices and transferred it in blocks using a DMA channel, similar to the GPS DMA channel. This allows up to 128-ARINC words to be transferred without interruption.

[7]MIL-SPEC MIL-N-49098 (EL), Navigational Set, Doppler AN/ASN-128( ), 26 July 1976.

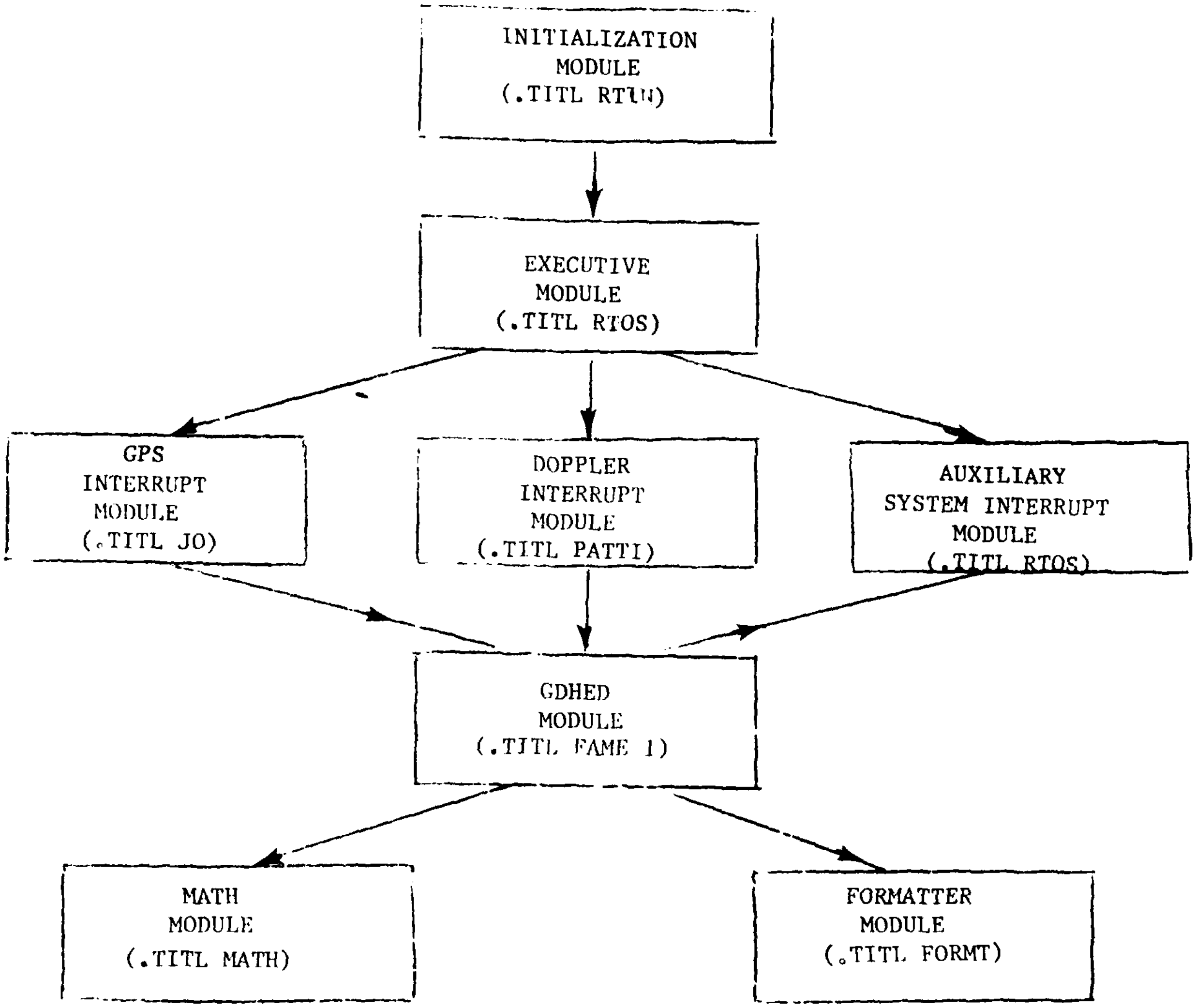


Figure 12. GDHED software modules

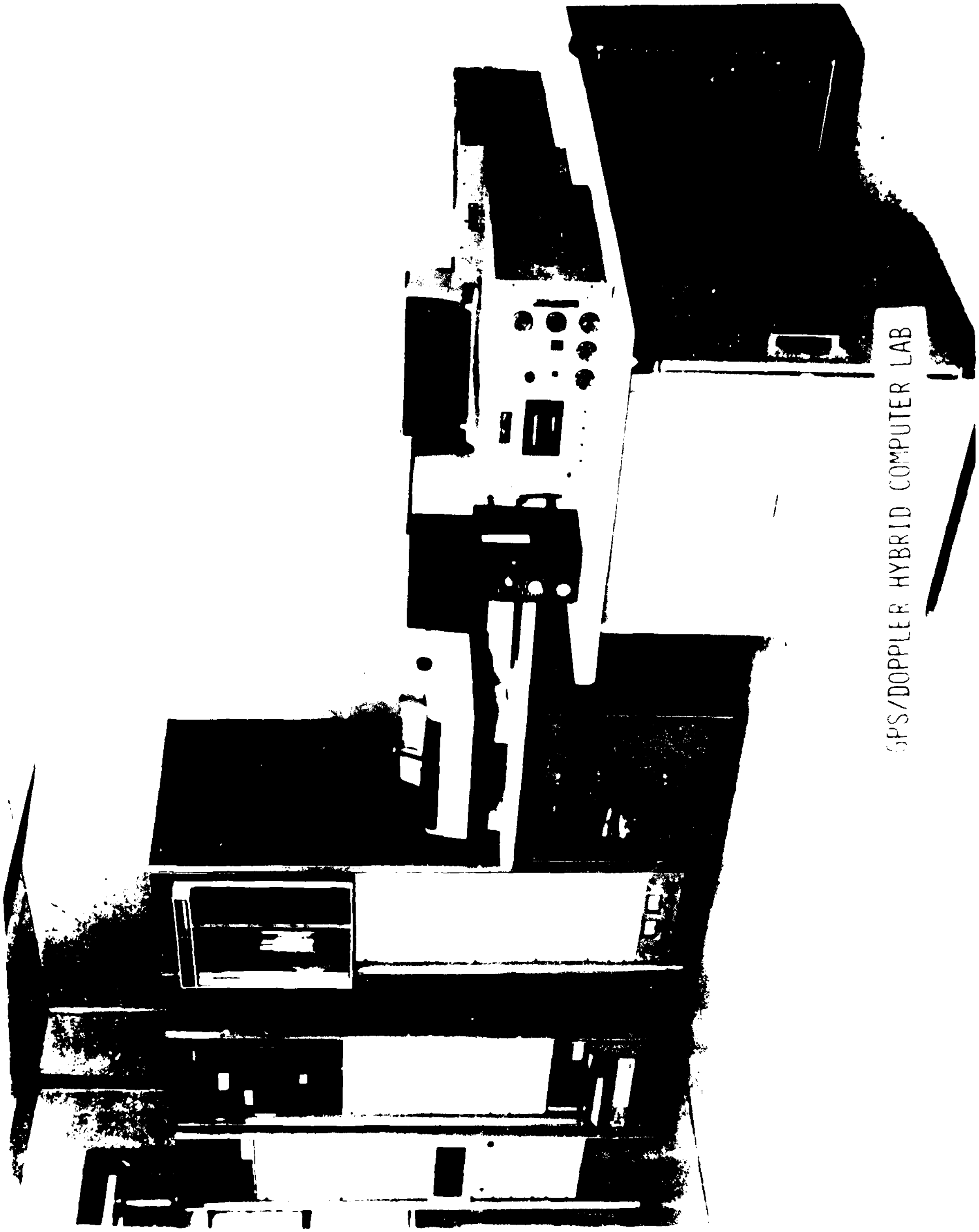


Figure 13. GPS/Doppler Hybrid Computer Laboratory

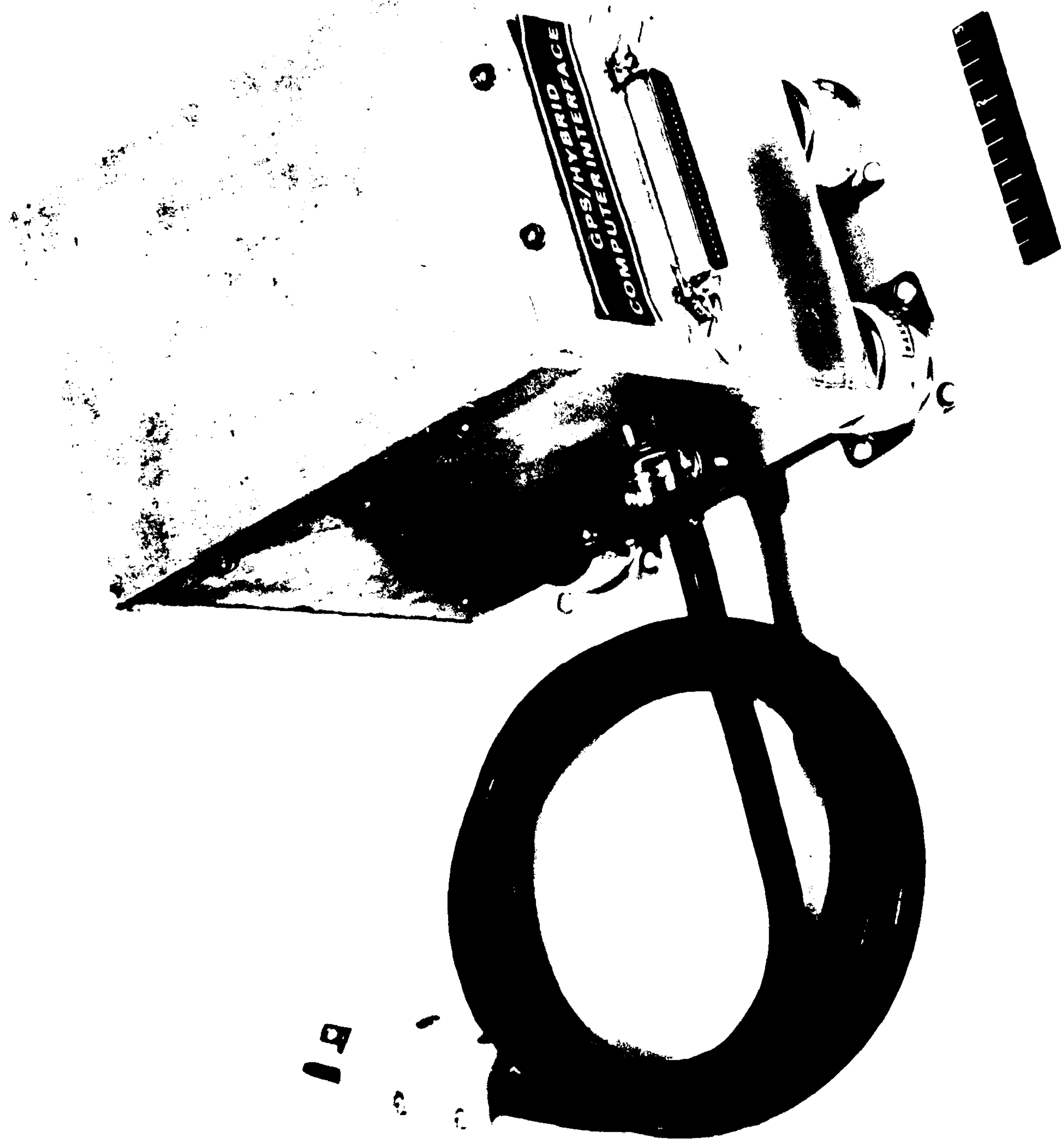


Figure 14. GPS/hybrid computer interface

Both the GPS and Doppler Units used DMA channels since this mode of transfer was not only the fastest (640 K words/sec) but also most efficient (block transfers). At the DMA speed, the HYBRID computer could easily accomodate the GPS data update rate (IIU block 206) of 0.64 seconds and the Doppler data update rate of 7.5/second.

The Doppler prototype interface unit as originally designed worked well; however, its size, weight, and power consumption was excessive. It was therefore decided to improve the design and incorporate its entire function within the HYBRID computer I/O chassis. The resultant saving in size, weight, and power is depicted in Figure 15.

d. Integration of Experimental GDHED System. The next step in the process was to perform as much hardware testing as possible without use of the computer. Then parts of the system were interconnected individually to the HYBRID Computer with simple programs loaded in computer memory. Gradually, this process was bootstrapped up until the entire system was functioning as designed.

The most difficult phase of this operation was to insure proper electromagnetic compatability between the subsystems. The proper type of interconnecting wire, shielding and sheathing, grounding and bonding, cable routing and cross-talk elimination in the interface cables was of paramount importance to assuring successful operation, since external and self-generated transients prevented the entire system from operating properly initially. Only when the above steps were completed was the hardware ready for integration testing.

e. Van Installation. The GDHED equipment was installed aboard this Activity's van (Figure 16) in preparation for van tests. This equipment is functionally depicted in Figure 17.

(1) Interior equipment. The installation design called for four racks of equipment that could be symmetrically located and provide for functional grouping of the equipment with good weight and balance properties.

Figure 18 shows the GPS and Doppler Equipment rack mounted. In the foreground we see the GPS equipment. In the background the Doppler equipment and a display mount housing the GPS and Doppler Control Display Units.

Figure 19 shows the Hybrid Computer Rack. Scanning from top to bottom we see the paper tape reader above, and the Hybrid Computer system with GPS/Computer Interface Unit below it.

Figure 20 shows the Teletype printer and magnetic cassette tape recorder; and below it the attitude reference system.

(2) Exterior equipment. The exterior equipment consisted of the respective GPS and Doppler antenna assemblies. Figure 21 shows the location of the GPS antenna on the van. This site was chosen to maximize signal reception and minimize superstructure shadowing and electromagnetic interference.

At the rear of the van, the Doppler antenna was mounted on a cantilever bracket running along the longitudinal axis of the van and extending horizontally from the van's roof; the Doppler antenna's aperture facing the earth's

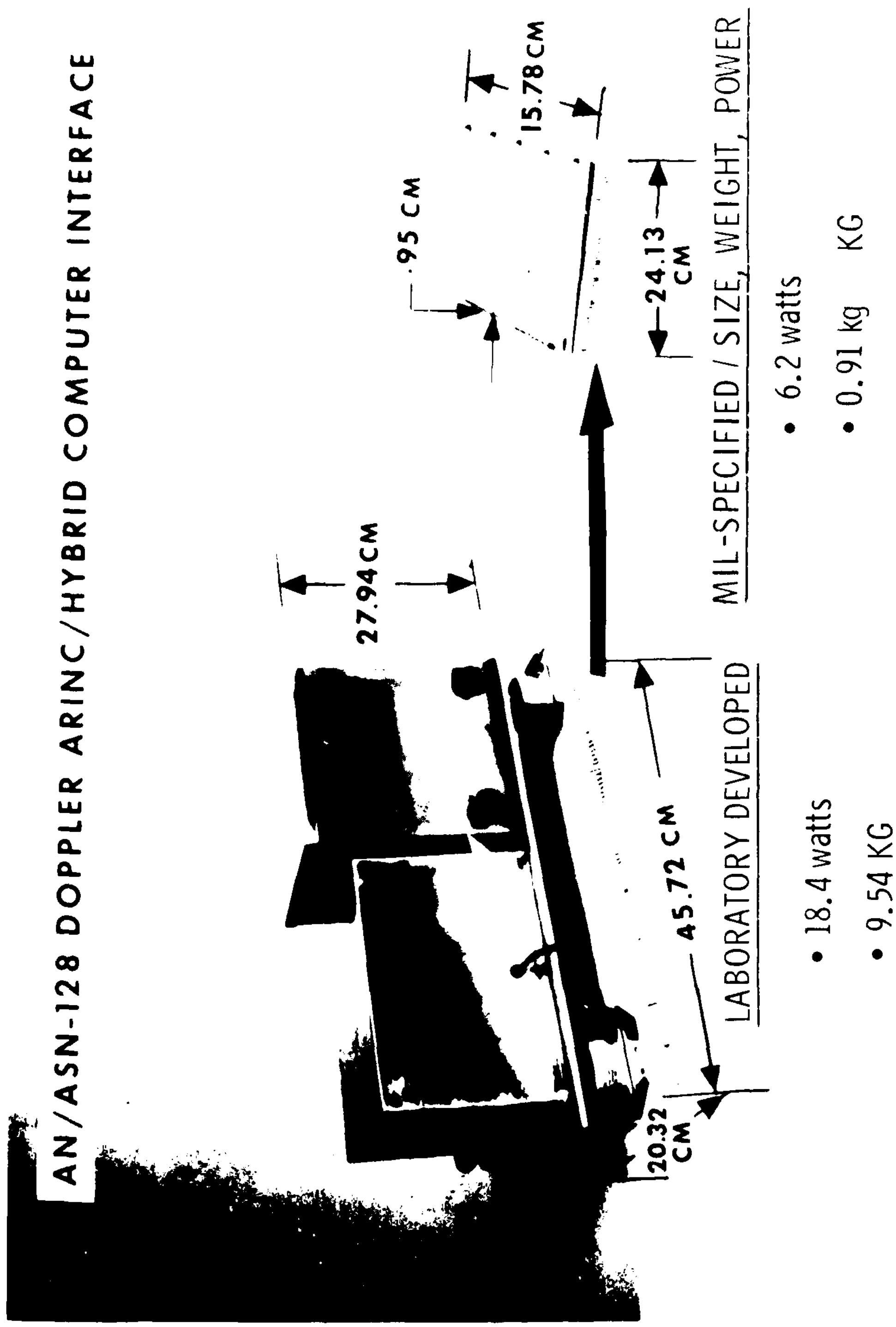


Figure 15. AN/ASN-128 Doppler/ARINC/hybrid computer interface

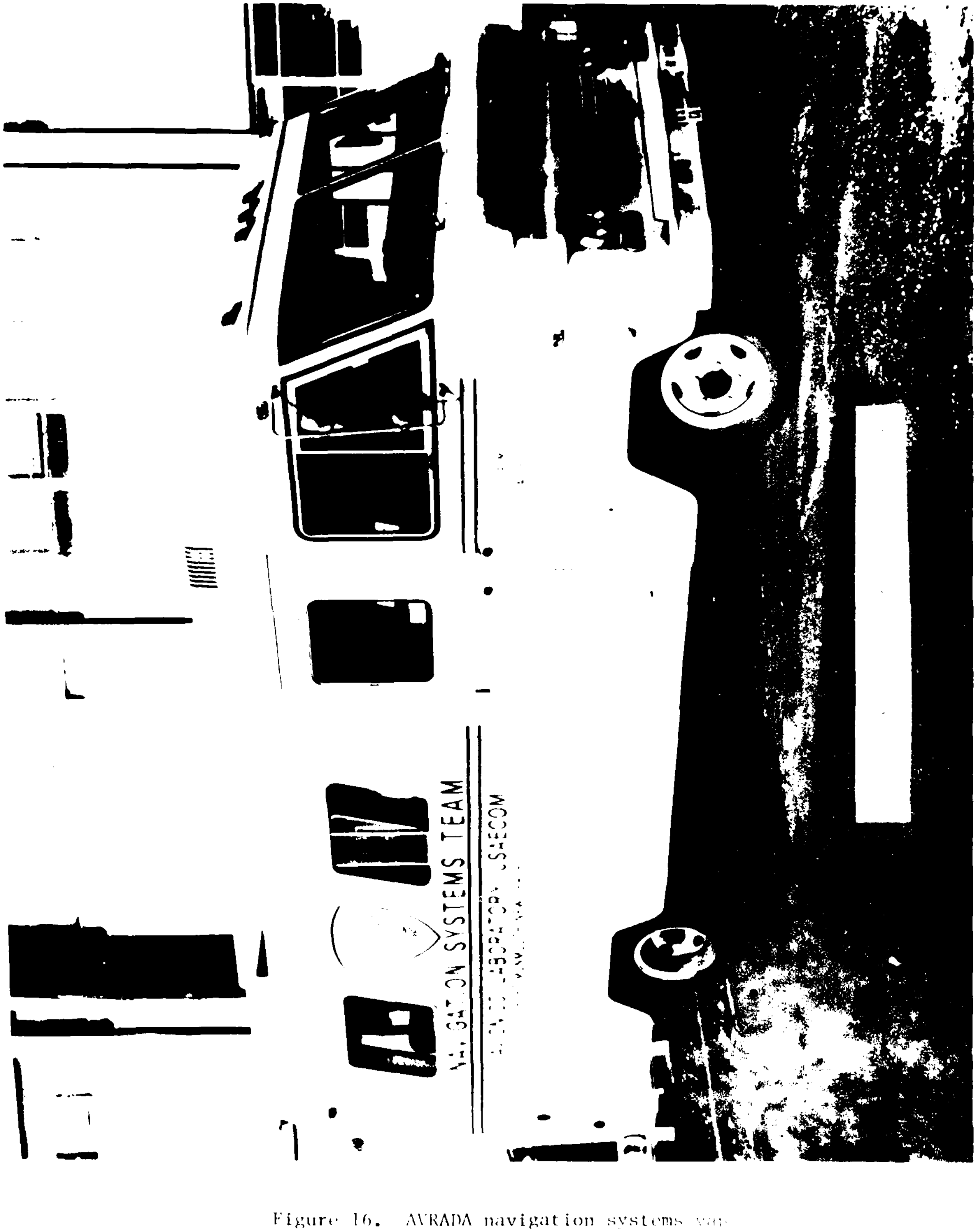


Figure 16. AVRADA navigation systems van

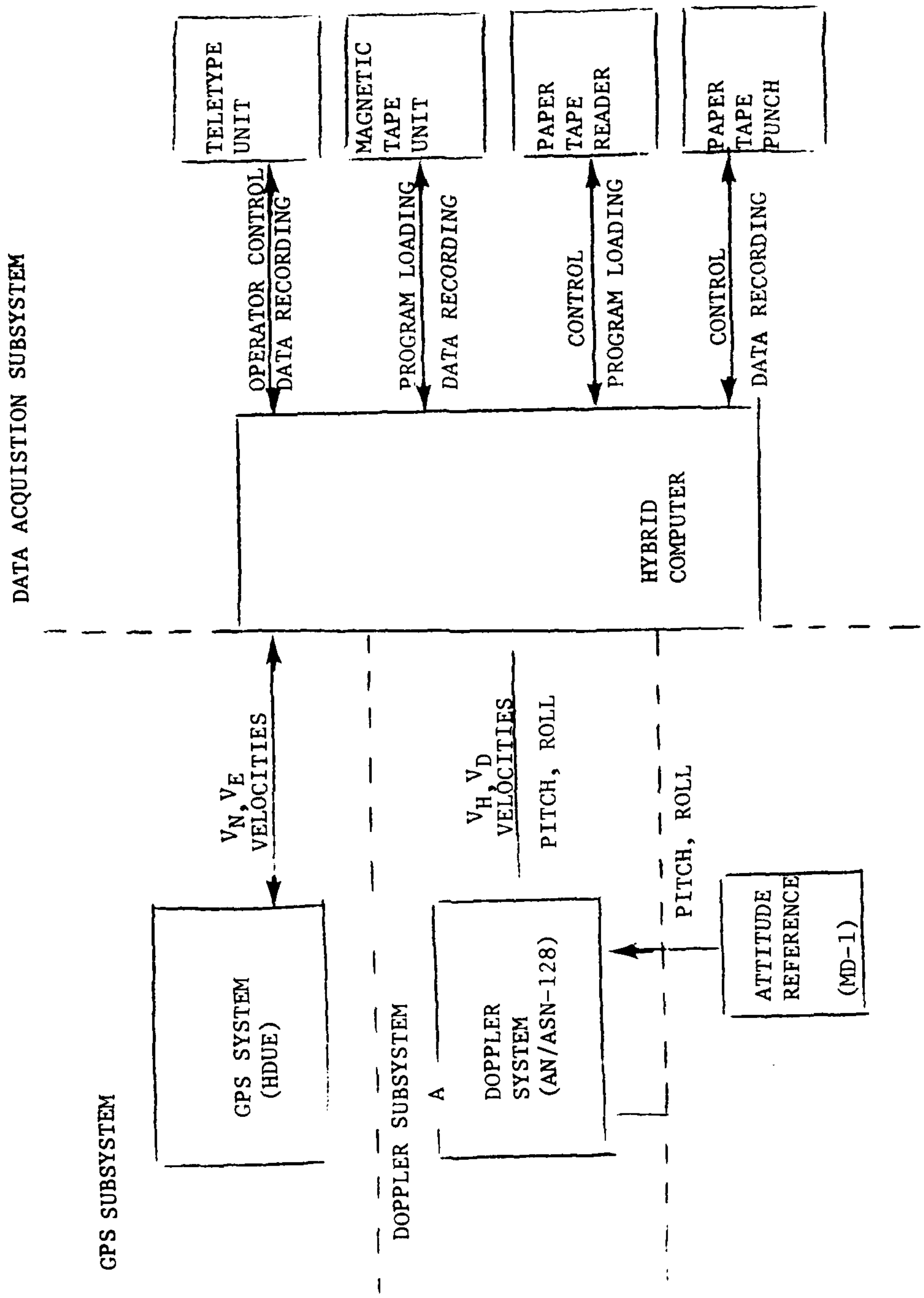


Figure 17. GDHED equipment functional flow diagram

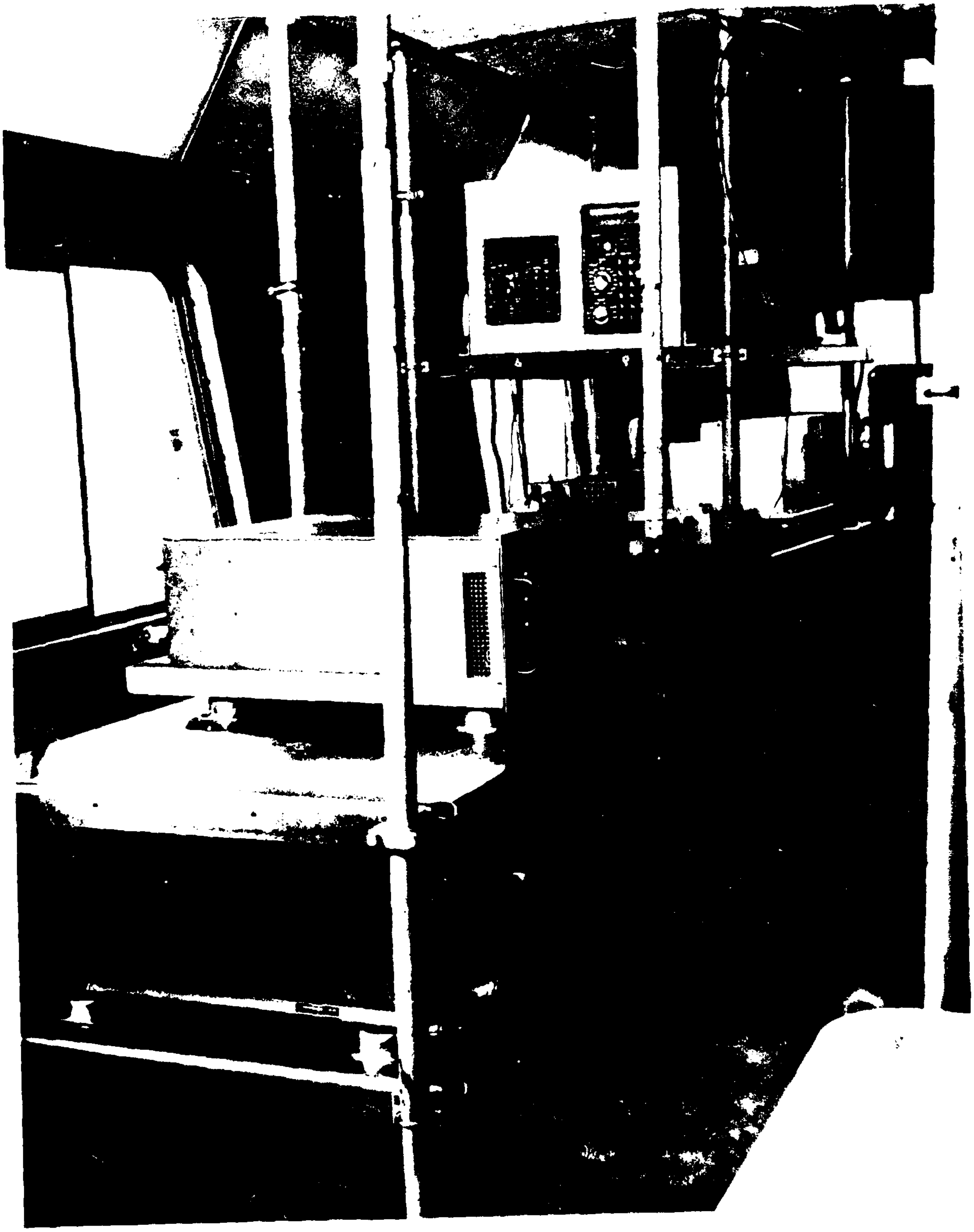
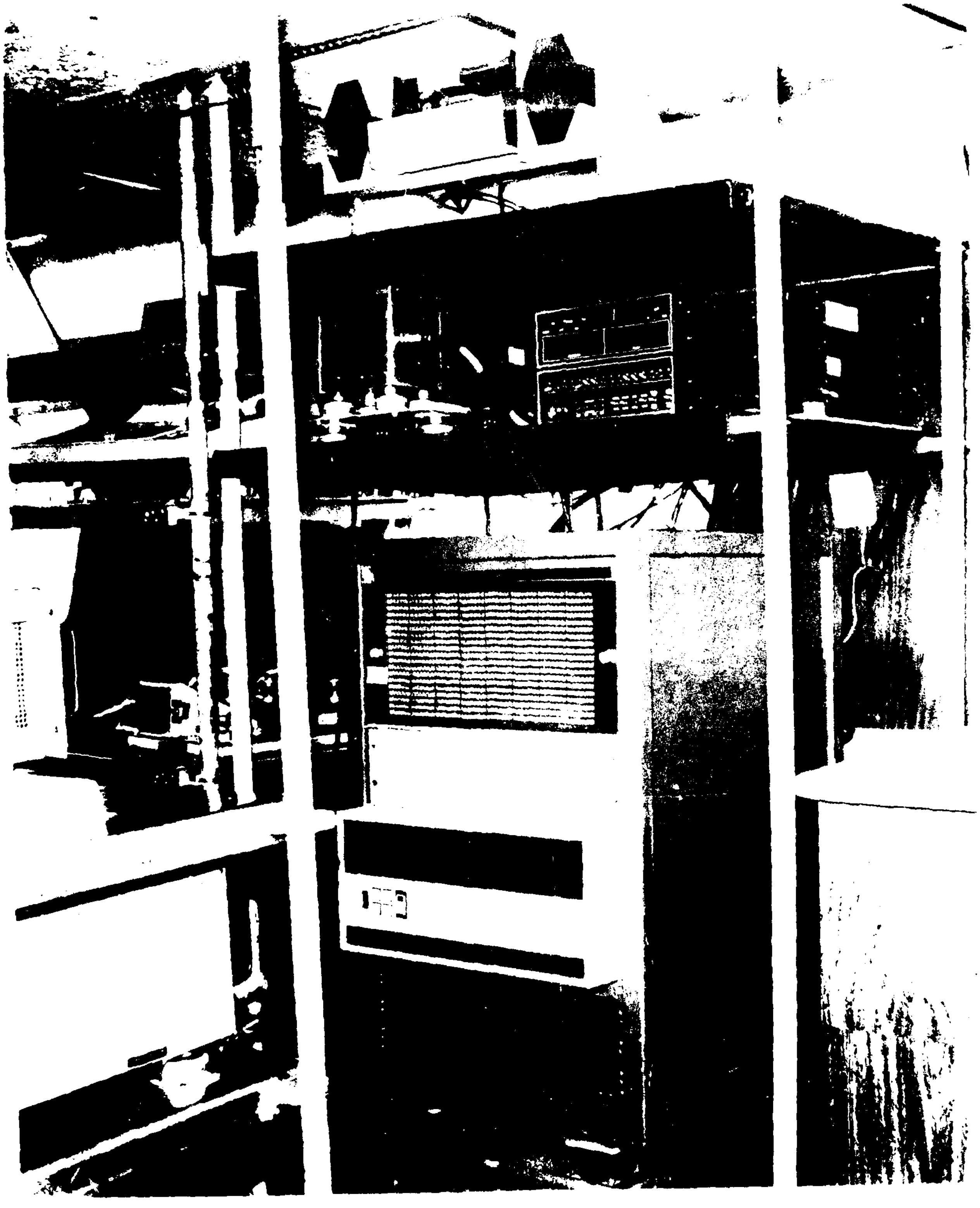


Figure 15. [illegible]



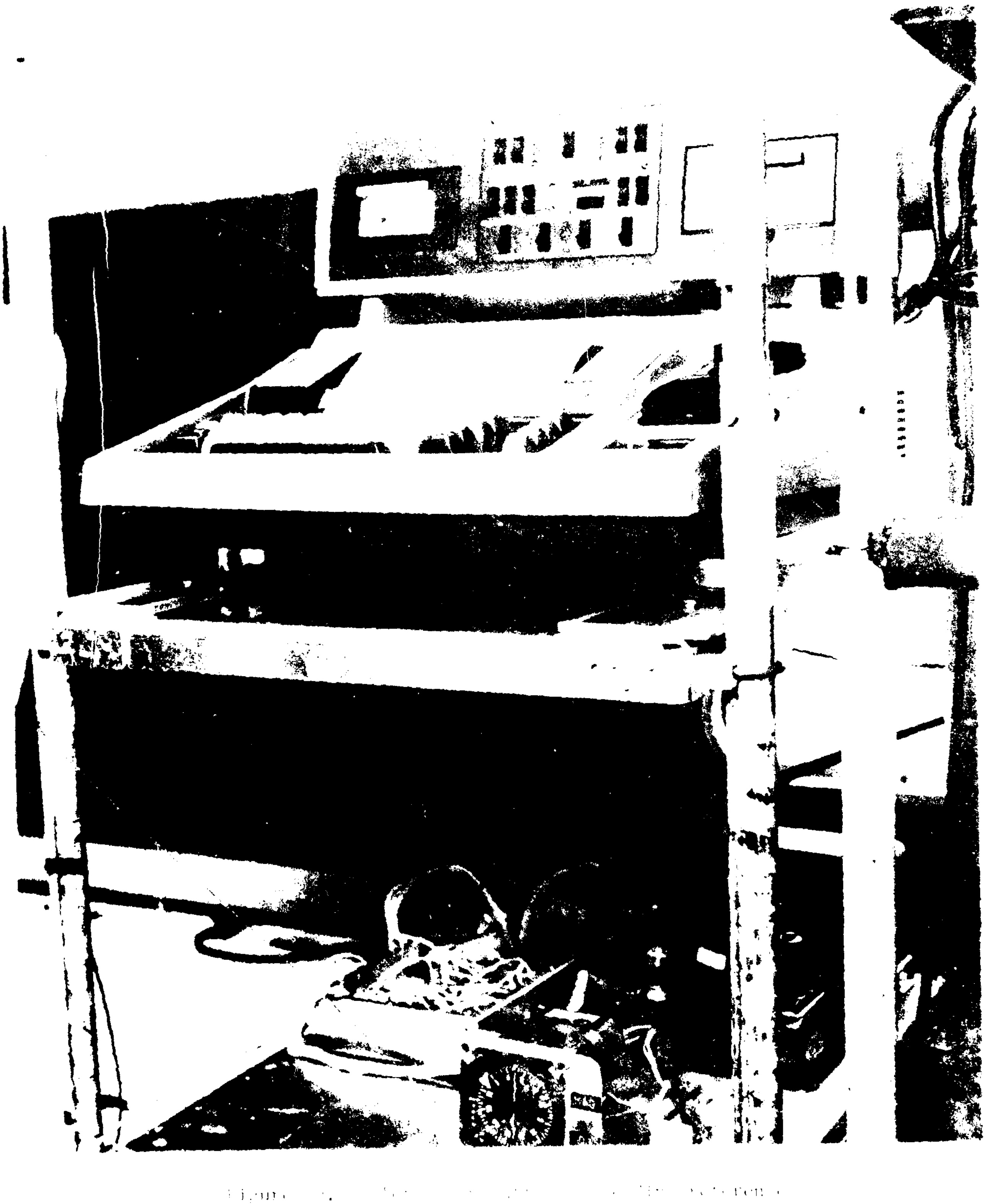


Figure [illegible]

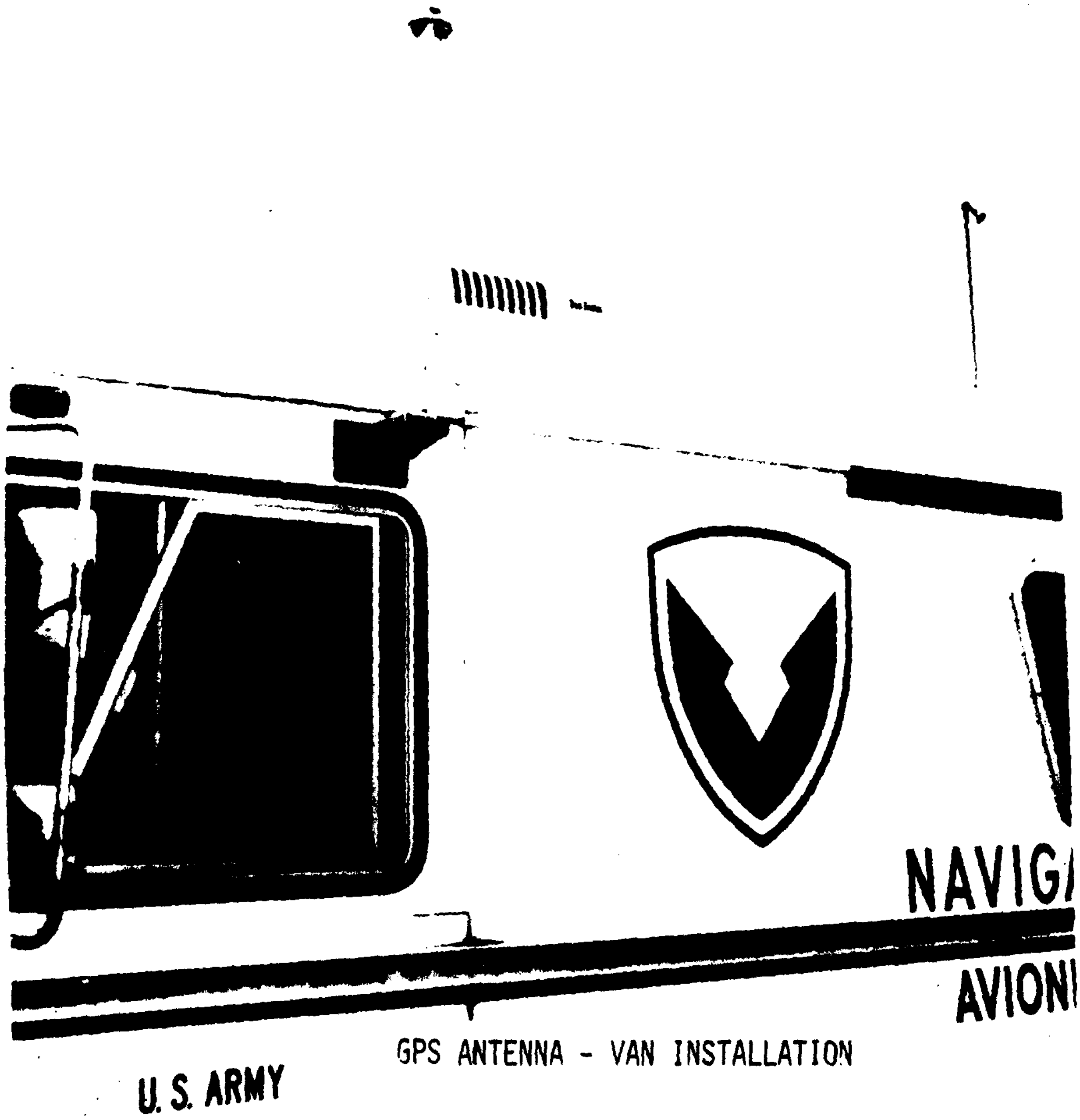


Figure 21. GPS antenna - van installation

surface (Figure 22). In accordance with the Doppler set's installation specification, the Doppler antenna was mounted such that no interference of the radiating beam occured due to any superstructure.[8]

It was important in mounting the Doppler antenna to avoid misalignment with respect to the longitudinal axis of the vehicle, since this condition would result in along-track (V_H) and cross track (V_D) velocity errors.

TABLE 5. GDHED SIZE, WEIGHT, AND POWER REQUIREMENTS

UNIT	SIZE H x W x D inches (CM)	WEIGHT lbs (kg)	POWER 115 V 400 Hz, 30 watts	POWER 115 V 60 Hz watts
GPS Antenna and Pre-amplifier	12 (30.5) antenna w/4 (10.2) dia base Prw amp 3.0 x 208.0 x 5.0 (7.6 x 20.3 x 12.7)	10 (22)	18.5 VDC @ 100 mz (1.85)	
GPS Equipment Rack	30.0 x 22.0 x 22.0 (76.12 x 55.82 x 55.82)	207 (455.4)	670	
AN/ASN-128 Doppler Antenna	14.56 x 13.48 x 1.92*	9 (19.8)	Supplied via Doppler SDC located in Doppler Equipment Rack	
AN/ASN-128 Doppler Rack	10 x 24 x 14 (25.37 x 60.90 x 35.53)	21 (46.2)	20 VDC @ 3.52 A (98.6)	
HYBRID Computer** Equipment Rack	12 x 17 x 19 (30.45 x 43.13 x 48.21)	50 (110)	500	1.2
Auxiliary Equipment Rack	36 x 22 x 22 (91.35 x 55.82 x 55.82)	75 (165)		250
	TOTAL	372 (818.4)	1170	351.65

*Does not include 5.58 x 5.58 x 2.91 (14.77 x 14.17 x 7.39) receiver-transmitter unit.
**Includes GPS/HYBRID computer Interface Unit

[8]MIL-I-59162 (EL), Military Specification, Navigational Set, Doppler AN/ASN-128, Installation and Acceptance Testing of.

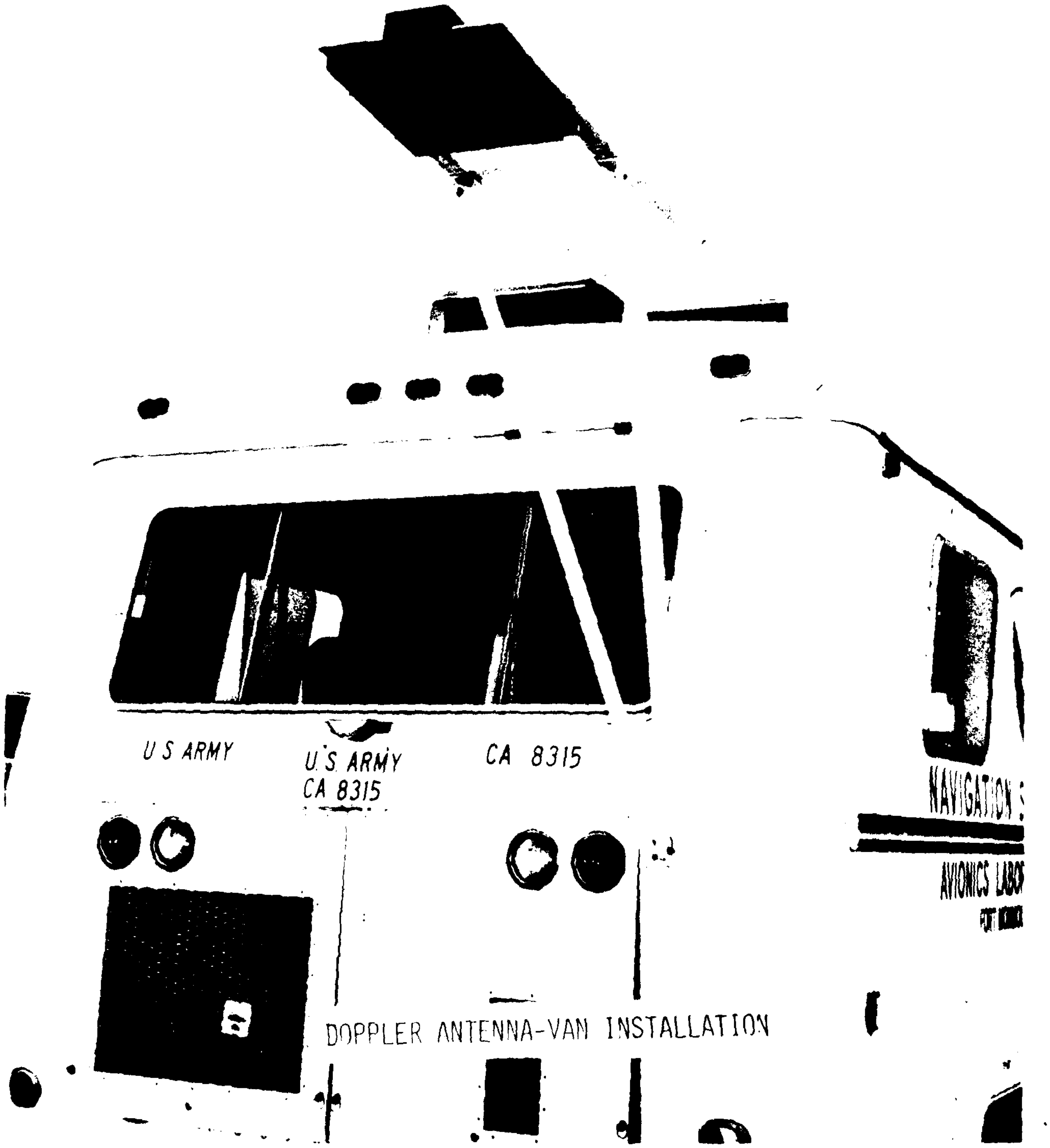


Figure 22. Doppler antenna - van installation

5. VAN TESTS

a. <u>Special Conditions: GPS Satellite Availability Considerations</u>. Although the final GPS satellite configuration will include 18 satellites providing continuous world-wide coverage, the satellite configuration during the time of the van tests included a total of 5 satellites, 4 of which are required to be in view at any one time to provide accurate velocity data. This limited satellite system configuration imposes time availability constraints in obtaining GPS data. Figure 23 provides a sample description of satellite availability for the Fort Monmouth, NJ area. For the purpose of planning and scheduling, acceptable continuous availability of GPS information was limited to a 2-hour time frame. Daily scheduling was centered about this time interval.

The van test was run 11 August 1980. For this date the following four GPS satellites became available in the Fort Monmouth, NJ area at 9:00 PM EDT: NAVSTARS 1, 3, 4, and 5. The calculated Geometric Dilution of Precision (GDOP) for this constellation over the Fort Monmouth area varied as follows:

$t_o \rightarrow \text{GDOP} = 4.8$

$t_o + 30 \rightarrow \text{GDOP} = 5.1$

$t_o + 60 \rightarrow \text{GDOP} = 6.9$

$t_o + 90 \rightarrow \text{GDOP} = 12.9$

$t_o + 120 \rightarrow \text{GDOP} = 40.1$

where t_o = inital time four satellites available (minutes)

The accuracy with which one can measure position/velocity and time is related to the accuracy in radial range measurement by this GDOP. When these satellites passed over Vandenberg AFB, CA, the measured range error for the satellites were as follows:

	RANGE ERROR (METERS)	
NAVSTAR	Mean	Std Deviation
1	-5.1	4.1
3	-2.4	-1.6
4	0.3	0.4
5	0.3	0.6

b. <u>Operating Area Characteristics</u>. Operating Area selection was based upon straightness (constant heading) of roads and open visibility to GPS satellites. In consideration of these factors, the following operating areas were chosen (Figure 24).

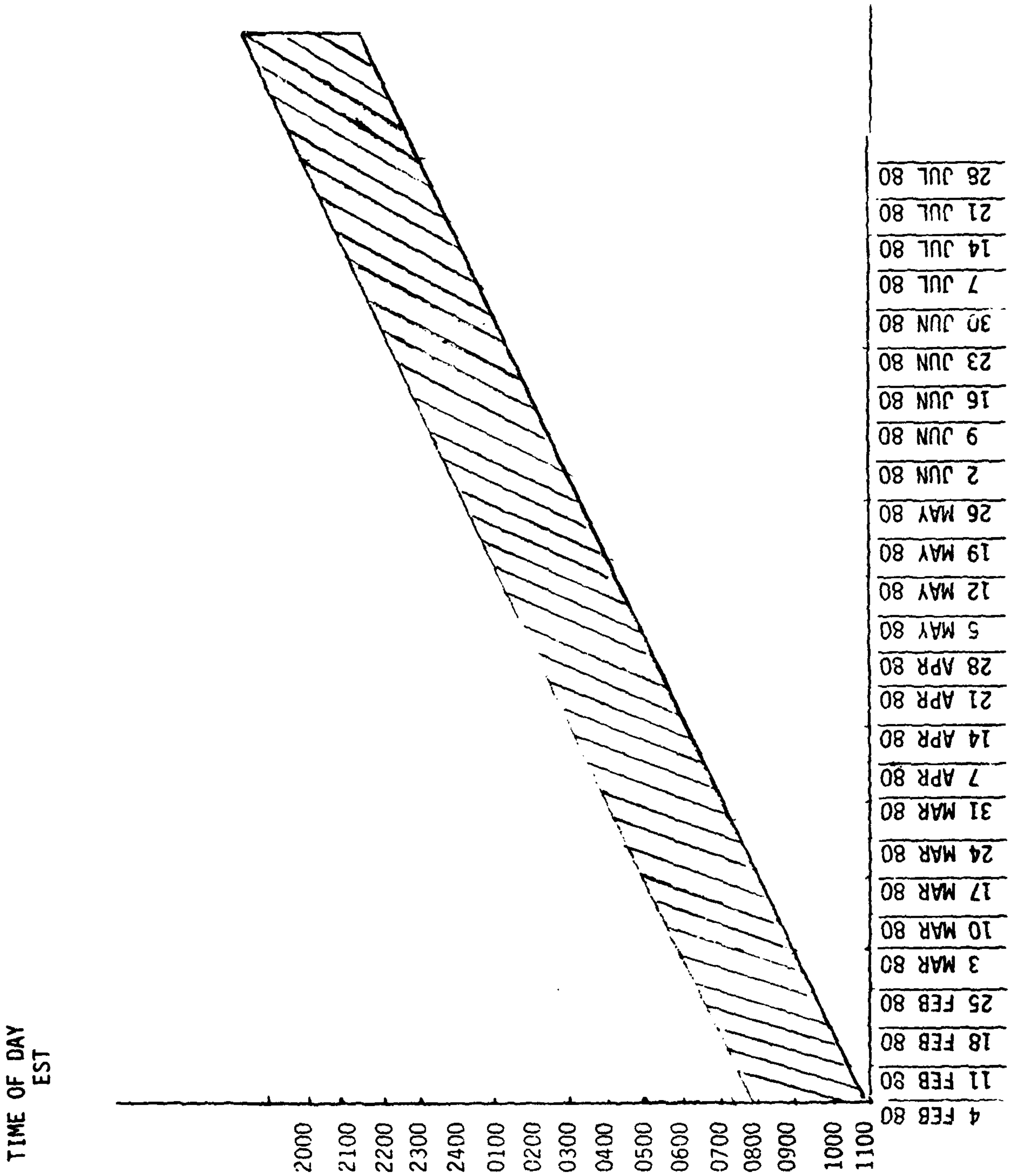


Figure 23. GPS availability at Fort Monmouth, NJ

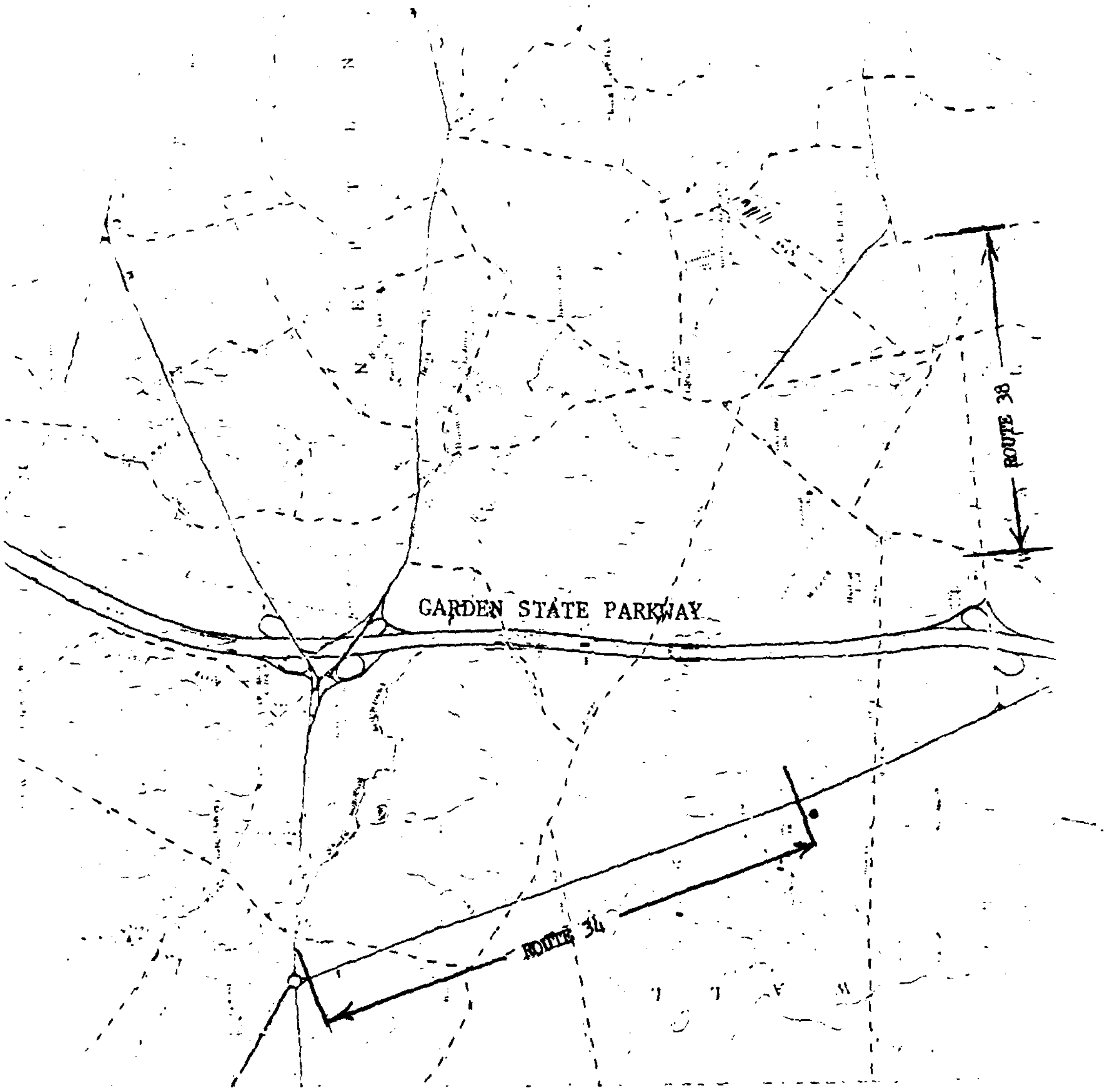


Figure 24. GDHED test site

(1) Route 34, between Collingwood Circle and Hurley's Pond Road, Monmouth County, NJ.

(2) Route 38, between Allenwood Road and Barkalow's Corner-Old Mill Road, Monmouth County, NJ. (Figure 23 was taken from a set of 1:24,000 U.S. Geological Survey maps for the New Jersey area.)

c. General Requirements. Operations consisted of a series of runs at various constant speeds up and down the selected operating sites. Table 6 summarizes the test run parameters.

TABLE 6. GDHED TEST PARAMETERS

RUN	ROUTE	HEADING (TRUE)	LENGTH (KM)	SPEED
1	38	87°	1.238	48.279 KM/HR (30 MPH)
2	38	267°	1.238	48.279 KM/HR
3	38	87°	1.238	32.186 KM/HR (20 MPH)
4	38	87°	1.238	32.186 KM/HR
5	34	342°	1.981	48.279 KM/HR
6	34	162°	1.981	48.279 KM/HR
7	34	342°	1.981	32.186 KM/HR
8	34	162°	1.981	32.186 KM/HR
9	34	342°	1.981	80.465 KM/HR (50 MPH)
10	34	162°	1.981	80.465 KM/HR

Ten runs were made utilizing velocity data from the GPS and Doppler systems and attitude data from the MD-1 gyro. The runs consist of two straight line legs. The first leg (Rt 38) is 1.238 km, the second leg (Rt 34) is 1.981 km.

The errors in each leg were determined by correlating the real-time HYBRID computer generated data with the "true" headings of the two legs obtained from 1:24000 U.S. Geological Survey maps. Figure 25 represents a sample printout by the onboard teletype of the real-time GDHED data.

The printout allowed the operator to know approximately how well the tests were progressing. The data was formatted such that the following information was made available:

TIME	GYRO	GDHED	ERROR	VELOCITY	GPS STATUS
5296.3	3.01	2.82	-0.19	82.2	4.0
5300.5	3.02	2.84	-0.18	88.7	4.0
5304.3	3.02	2.84	-0.18	86.4	4.0
5308.0	3.01	2.82	-0.19	83.4	4.0
5311.8	3.02	2.79	-0.23	78.2	4.0
5315.6	3.02	2.79	-0.23	76.4	4.0
5319.3	3.04	2.81	-0.23	81.6	4.0
5323.1	3.04	2.77	-0.27	83.6	4.0
5326.8	3.04	2.83	-0.21	86.7	2.0
5330.5	3.03	2.81	-0.22	87.2	2.0
5334.3	3.03	2.83	-0.20	85.8	3.0
5337.4	3.04	2.82	-0.22	82.8	3.0
5341.1	3.05	2.85	-0.20	81.3	4.0
5353.1	3.04	2.80	-0.24	82.2	3.0
5360.6	3.05	2.86	-0.19	87.6	4.0
5374.9	3.06	2.83	-0.22	79.9	3.0
5379.3	3.06	2.79	-0.27	83.7	3.0
5383.0	3.07	2.85	-0.21	87.5	4.0
5386.8	3.06	2.81	-0.25	84.0	4.0
5390.6	3.05	2.82	-0.23	86.3	4.0
5394.3	3.05	2.83	-0.22	87.3	4.0
5397.4	3.07	2.85	-0.21	85.1	4.0
5401.2	3.07	2.83	-0.24	85.8	4.0
5405.5	3.07	2.80	-0.27	78.5	4.0
5409.3	3.07	2.85	-0.22	81.9	4.0
5413.0	3.07	2.85	-0.22	87.2	4.0
5420.5	3.07	2.78	-0.28	81.1	4.0

Figure 25. Real-time GDHED sample data listing

(1) GDHED data sample time (in seconds)

(2) "Quick-look" gyro heading

(3) GDHED heading

(4) "Quick-look" GDHED heading error

(5) GPS satellite status

d. Test Results Analysis. The GDHED van tests were 2 hours in duration and began after the GPS set locked on and acquired four GPS satellites. A summary of heading accuracy determined for the series of runs is shown in Table 7.

TABLE 7. GDHED HEADING ACCURACY TEST RESULTS

RUNS	HEADING ERROR (DEGREES)		TOTAL VEHICLE VELOCITY KM/HR)		# OF READINGS
	MEAN	STD DEVIATION	MEAN	STD DEVIATION	
1	0.28	1.65	54.47	2.39	27
2	0.52	1.60	53.94	4.05	37
3	0.53	1.73	37.11	2.09	47
4	0.36	1.64	36.52	3.19	62
5	-0.44	1.56	53.28	4,45	40
6	-0.37	1.37	51.21	2.50	41
7	-1.12	2.61	36.45	2.02	76
8	-0.00	2.00	35.73	2.29	72
9	0.14	1.02	83.34	3.02	26
10	-0.36	1.38	83.79	3.26	27
TOTAL	-0.11	1.90			455

Analysis of the limited GDHED test results successfully demonstrates that a new means of determining heading of a vehicle now exists in addition to the present magnetic and inertial mechanizations. The cumulative GDHED system mean heading error of -0.11 degrees and standard deviation of 1.90 degrees falls short of the AN/ASN-128 Doppler heading accuracy requirement of 1 degree standard deviation (magnetic mode).

The GDHED performance fell short of expectations (GDHED error analysis predicted 0.371-degree standard deviation) mainly because of the poor GDOP of GPS, and uncompensated clock drifts of the GPS satellites following update over Vandenberg, AFB and the time they are available at Fort Monmouth, NJ. Even under the best conditions (namely, during the GPS Phase I Field Tests at Yuma Proving Grounds, AZ), the GPS velocity error (50th percentile) was only 0.3 meters/second (0.582 knots).[9] Although individual GPS and Doppler velocities were not recorded during the van tests, based upon the test results and error analysis (refer to Table 4), it is estimated that GPS velocity errors were in the order of 2.0 knots. Therefore, it is expected that with better GPS velocity accuracy, and improved GDOP, the more accurate the GDHED system.

An additional factor that added to the GDHED error, as indicated in the last column of the sample GDHED real-time data printout (Figure 25), was the intermittent loss of one or more of the GPS satellites during the runs, resulting in a degraded mode of GPS performance.

6. CONCLUSIONS

a. An accurate heading reference for airborne and ground vehicles can be generated by combining Earth-referenced velocities derived from GPS with Body referenced velocities derived by the AN/ASN-128 Doppler.

b. The accuracy of the GDHED system as presently configured (approx -0.1° mean error, 1.9° rms standard deviation) must be improved to meet the AN/ASN-128 Doppler's Heading accuracy requirement.

7. RECOMMENDATIONS

a. Further van and flight tests should be run to verify performance of the GDHED system under varying dynamic conditions.

b. The GDHED algorithms should be optimized via Kalman filtering and improvement verified via van and flight tests.

c. The GDHED algorithms should be applied to a Strapdown accelerometer, integrated velocity sensor/GPS system, to avoid use of active radiators in the system.

[9]Final User Field Test Report for the NAVSTAR (Global Positioning System Phase 1, GPS-GD-025-C-US-7708, DATA ITEM A01K, General Dynamics, 25 June 1979.

8. ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Mr. John Gratola, Ext. Ref Navigation Branch for his outstanding performance throughout the development of this project, and without whose support this project would not have been completed on schedule.

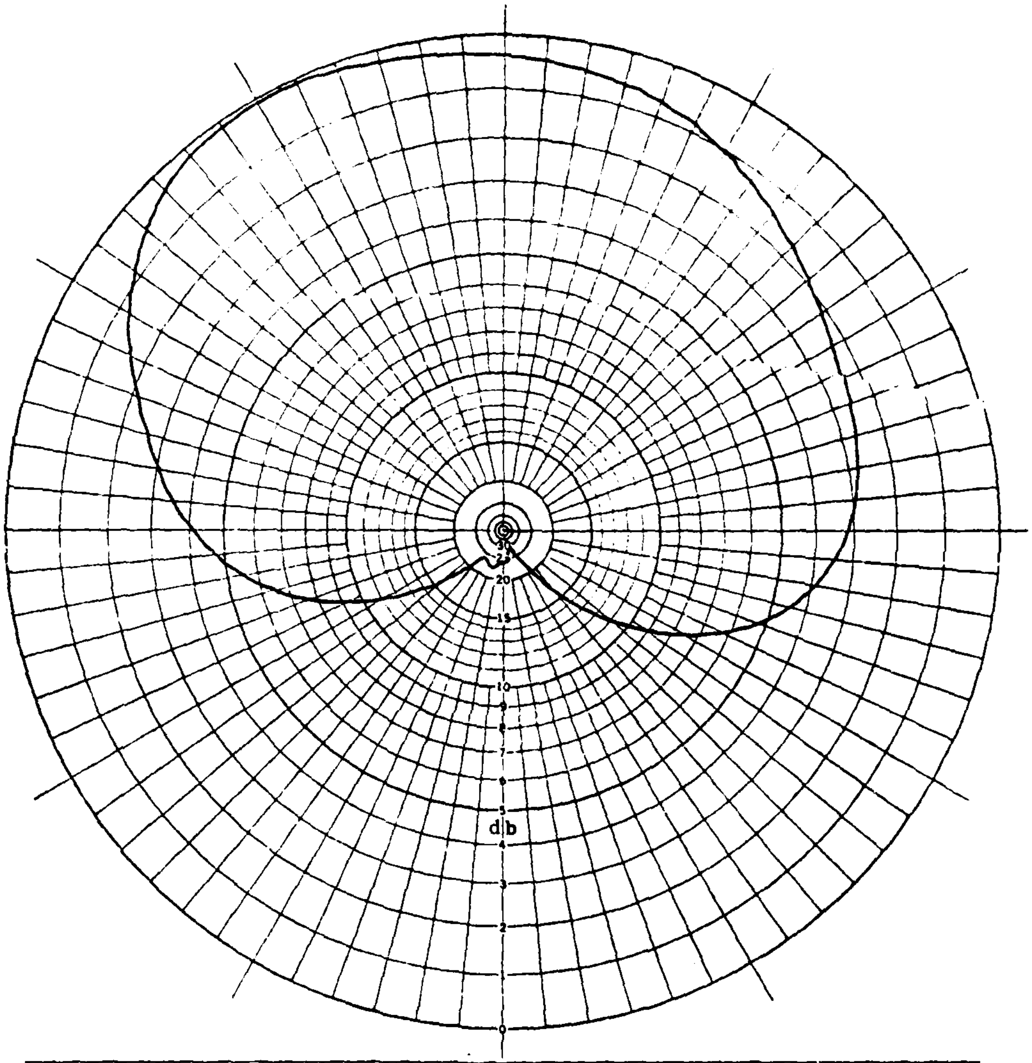
Special recognition is in order for Mr. Walter Weiss, RAYCOM Industries, for his contribution in the design and fabrication of the GPS Interface Unit.

Special thanks are in order for Mr. John Medea, ERADCOM, Electronics Warfare Laboratory, for his technical assistance in developing the GDHED software.

Finally, the technical service provided by Ms. Maureen Amos in running the computer simulations of the GDHED error analysis is gratefully acknowledged.

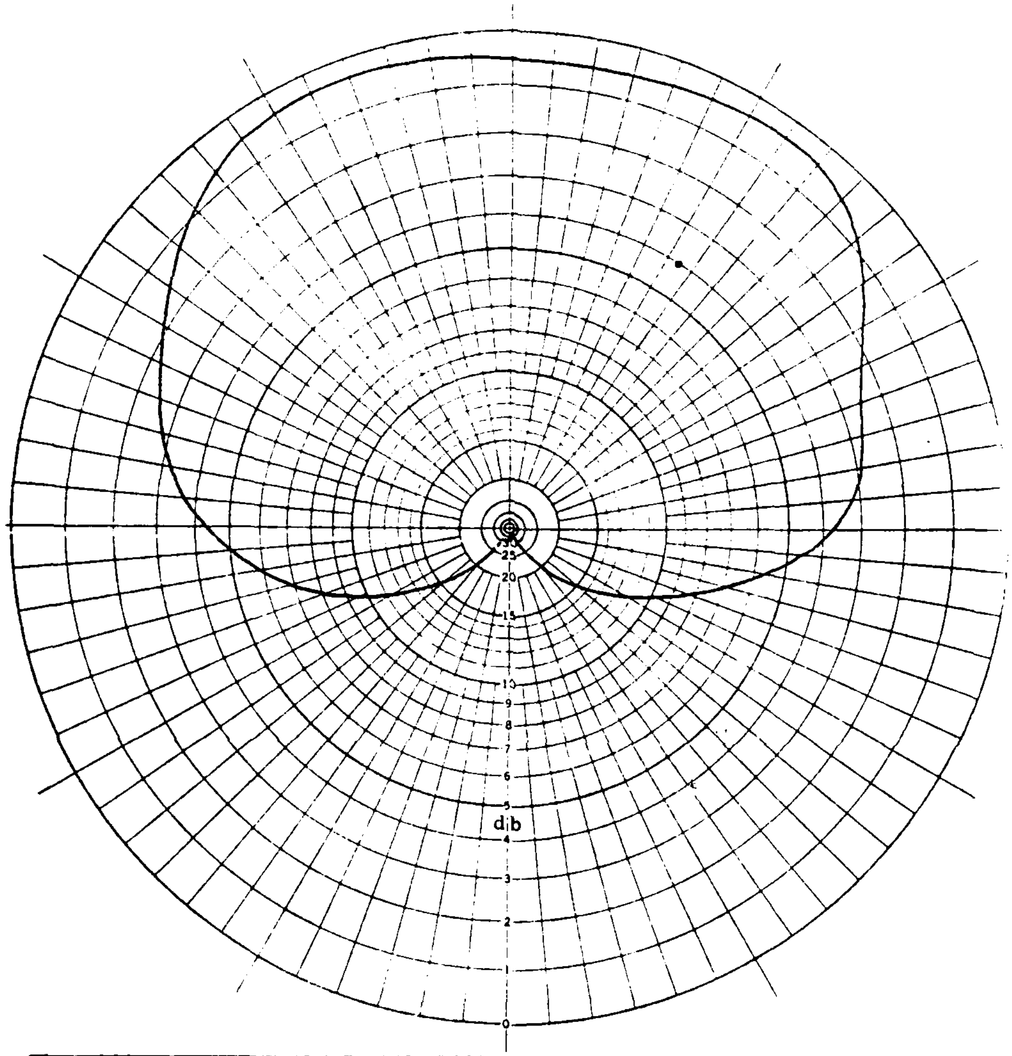
APPENDIX A

GPS ANTENNA PATTERNS

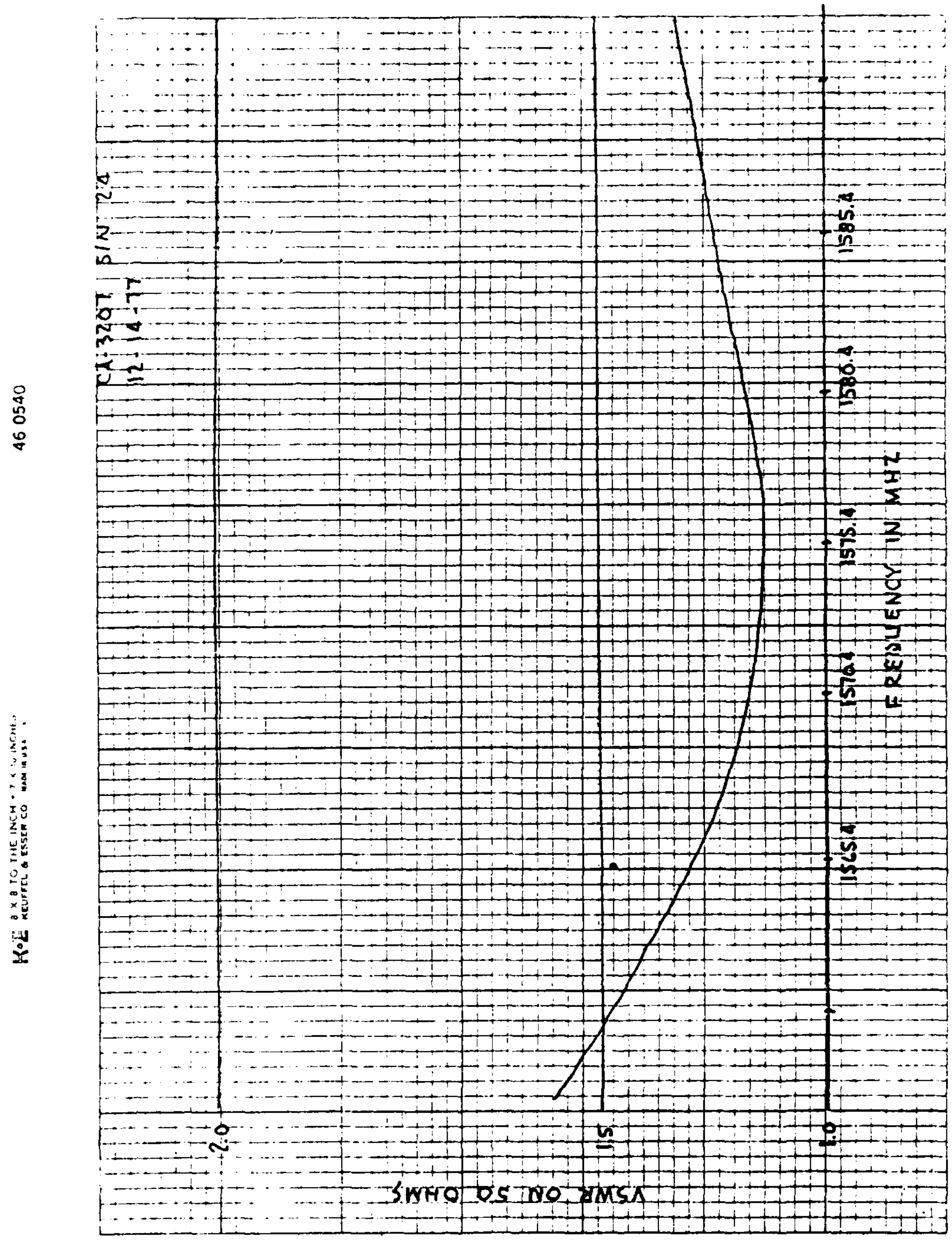


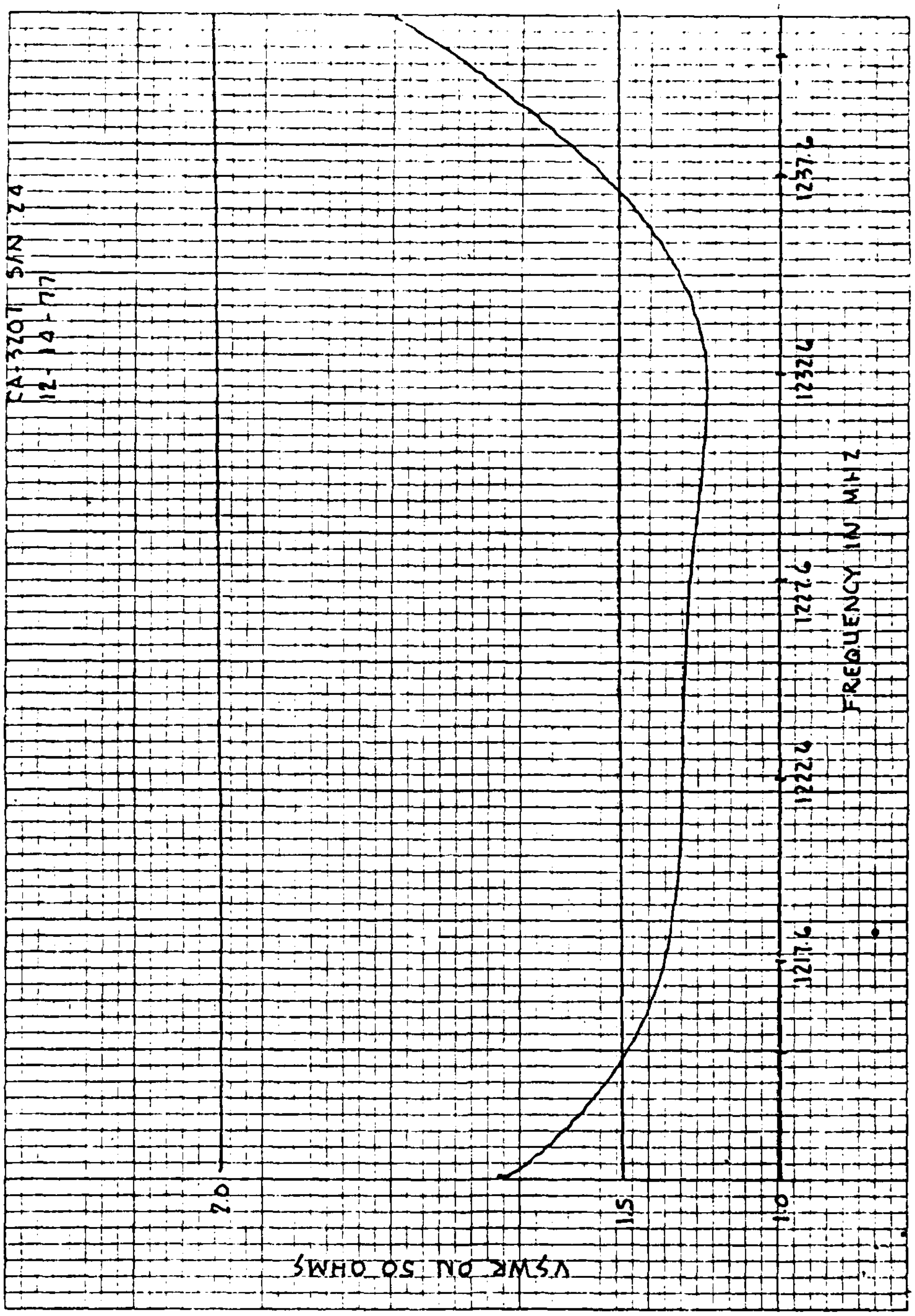
ANTENNA: CA-3207 S/N 24	PATTERN CUT IN:
FREQUENCY: 1227.6 MHZ	AZIMUTH PLANE AT ____° ELEVATION
PLOTTED IN:	ZENITH PLANE AT 0 ° RELATIVE AZIMUTH
VOLTAGE * POWER ____	POLARIZATION: E_θ ____ RHC *
DATE: 12-14-77	E_ϕ ____ LHC ____

CHU ASSOCIATES



ANTENNA: CA-3207 S/N 24	PATTERN CUT IN:
FREQUENCY: 1575.4 MHz	AZIMUTH PLANE AT ____ ° ELEVATION
PLOTTED IN:	ZENITH PLANE AT 0 ° RELATIVE AZIMUTH
VOLTAGE * POWER ____	POLARIZATION: E_θ ____ RHC *
DATE: 12-14-77	E_Φ ____ LHC ____


CA-3207 S/N 24
12-14-77
VSWR ON 50 OHMS
2.0
1.5
1.0
FREQUENCY IN MHZ



CA-3207 S/N 24
12-14-77
VSWR ON 50 OHMS
2.0
1.5
1.0
1217.6
1222.6
1227.6
1232.6
1237.6
FREQUENCY IN MHZ

APPENDIX B

GPS PREAMPLIFIER CHARACTERISTICS

SET X PRE-AMPLIFIER CHARACTERISTICS

A.	I/O CONNECTOR TYPE	N
B.	NUMBER OF ANTENNA INPUTS	1
C.	NUMBER OF SIGNAL/POWER CONNECTORS	1
D.	NUMBER OF CALIBRATION SIGNAL INPUTS	1
E.	CENTER FREQUENCIES	L_1/L_2
F.	IMPEDANCE	50 OHMS
G.	VSWR (MAX)	2.5:1
H.	GAIN (RELATIVE TO 3 db BANDWIDTH)	27 ± 3 dB
I.	INPUT SIGNAL	-50 TO -180 dBv
J.	BURNOUT PROTECTION	0 dBv
K.	BANDWIDTH (70 dB)	±55 MHz
L.	BANDWIDTH (3 dB)	±18 MHz
M.	NOISE FIGURE (MAX)	3.75
N.	CALIBRATION SIGNAL LEVEL	-42 dBv
O.	POWER REQUIREMENTS	18.5 vdc @ 100 ma
P.	GROUP DELAY VARIATIONS (MAX)	10 msec
Q.	L_1/L_2 ISOLATION (MIN)	50 dB
R.	PREAMPLIFIER 1 dB COMPRESSION POINT	+10 dBm
S.	CALIBRATION SIGNAL OUTPUT	-95 dBv
T.	CALIBRATION FREQUENCIES	34f, θ PRN $274t_f$ (f_c - 5.115 MHz)

APPENDIX C

GDHED PROGRAM LISTING (ERROR ANALYSIS)

```
      PROGRAM LA(INPUT,OUTPUT)
      DIMENSION A(7)
      B=0.
      D=0.
      N=0
      DO 31 K=10,100,10
      UT=K
      UX=.57735027*UT
      UY=UX
      UZ=UY
      DO 20 J=10,360,10
      H1=.01745329*J
      DO 15 L=1,31,5
      XP=.01745329*L
      DO 10 M=1,46,5
      R=.01745329*M
      N=N+1
      XPHI=0.
      XH=H1
      UH=(COS(XP))*(UX)+(SIN(XP))*(SIN(R))*(UY)+
     8(SIN(XP))*(COS(R))*(UZ)
      UD=(COS(R))*UY-(SIN(R))*UZ
      UN=(COS(XH))*UH-(SIN(XH))*UD
      UE=(COS(XH))*UD+(SIN(XH))*UH
```

```
      UDP=XPHI
      UDR=(-UY*SIN(R))-(UZ*COS(R))
      UDUX=XPHI
      UDUY=-SIN(R)
      UDUZ=-(UZ*COS(R))
      UHUZ=(SIN(XP)*COS(R))
      UHUY=(SIN(XP)*SIN(R))
      UHUX=COS(XP)
      UHR=((COS(R)*SIN(XP))*UY)+((-SIN(R)*SIN(XP))*UZ)
      UHP=(-SIN(XP)*UX)+((COS(XP)*SIN(R))*UY)+((COS(XP)*COS(R))
     S*UZ)
      XK=(1.)/(1.+(((UH*VE)-(UN*UD))/((UN*UH)+(UD*VE)))**2)
      D=(((UN*UH)+(UD*VE))**2)
      HUN=XK*((-VE*(UH**2+UD**2))/(D))
      HUD=XK*((-UH*(VE**2+UN**2))/(D))
      HVE=XK*((UN*(UH**2+UD**2))/(D))
      HUH=XK*((UD*(VE**2+UN**2))/(D))
      DO 3 I=1,7
      CALL BOXNO(RND1,RND2)
      A(I)=RND1
      D1=A(1)
      D2=A(2)
      D3=A(3)
      D4=A(4)
      D5=A(5)
      D6=A(6)
```

```
      D7=A(7)
3     CONTINUE
      DVE=.1*D1
      DVN=.1*D2
      DP=.05235988*D3
      DR=.05235988*D4
      DVX=((.0025*VT)+.1)*D5
      DVY=((.0025*VT)+.1)*D6
      DVZ=((.001*VT)+.05)*D7
      DVH=(VHP*DP)+(VHR*DR)+(VHVX*DVX)+(VHVY*DVY)+(VHVZ*DVZ)
      DVD=(VDP*DP)+(VDR*DR)+(VDVX*DVX)+(VDVY*DVY)+(VDVZ*DVZ)
      EH=(HVH*DVH)+(HVE*DVE)+(HVD*DVD)+(HVN*DVN)
      B=B+EH
      G3=(EH-(.0007858796))**2
      E=E+G3
10    CONTINUE
15    CONTINUE
20    CONTINUE
31    CONTINUE
      AVG=B/N
38    FORMAT(1X,"THE MEAN ERROR IS=",F20.10)
      PRINT 38,AVG
39    FORMAT(1X,"N=",I10)
      PRINT 39, N
      F=(.0000277785)*E
46    FORMAT(1X,"THE STANDARD DEVIATION IS=",F20.10)
      PRINT 46, F
      END
```

APPENDIX D

GDHED RTOS VAN TEST SOFTWARE

```
;*****************************************************************;
;                                                                 ;
;    EEEEEEE    X       X     EEEEEEE     CCCCCCC                 ;
;    E           X     X      E           C                       ;
;    E            X   X       E           C                       ;
;    EEEEEEE       XX         EEEEEEE     C                       ;
;    E            X   X       E           C                       ;
;    E           X     X      E           C                       ;
;    EEEEEEE    X       X     EEEEEEE     CCCCCCC                 ;
;                                                                 ;
;*****************************************************************;
;
;               PROJECT  GPS/DOPPLER HYBRID NAVIGATION SYSTEM
;
;               PROJECT ENGINEER: JACK GRAY
;
;                 * EXECUTIVE PROGRAM "EXEC" *
;
;SYSTEM DEFINITIONS
;
                                                        ;
 DUSR JOBS=32
               ;ALLOWED IN THE SYSTEM
 DUSR   CHAN=2  ;NUMBER OF  XMIT/ RCV CHANNELS
 DUSR TTYS=1
 DUSR FREQ=1
               ;0=LINE ,1=10 HZ ,2=100 HZ ,3=1000HZ.
 DUSR SHALT=0   ;SYSTEM RESOURCES DEPLETED PARAMETER
               ;0=> HALT; 1=> JUMP TO USER SUPPLIED
               ;PROGRAM WITH ENTRY POINT ". SHLT"
 DUSR SMON=30   ;SYSTEM MONITOR
                                                        ;
;DEVICE/OPTION DEFINITIONS --
;       0 => NOT AVAILABLE
;       1 => DEVICE ON SYSTEM
                                                        ;
 DUSR   PWRFL=1 ;POWER MONITOR/AUTO RESTART
 DUSR HSR=1
 DUSR HSP=1
 DUSR PRINT=1
 DUSR   PLOT=0  ;INCREMENTAL PLOTTER
 DUSR   CARD=0  ;CARD READER
 DUSR   DISK=0  ;DISK (FIXED HEAD)
 DUSR   A2D=0   ;ANALOG TO DIGITAL CONVERTER
 DUSR   TAPE=1  ;MAGNETIC TAPE UNIT
 DUSR   DCOM=0  ;DATA COMMUMNICATIONS MULTIPLEXER
 DUSR   QMUX=0  ;TYPE 4060 ASYNCHRONOUS MULTIPLEXOR
 DUSR   IBM=0   ;TYPE 4025 NOVA--SYSTEM 360 INTERFACE
 DUSR   DPACK=0         ;MOVING HEAD DISK HANDLER
 DUSR   SYNC=0  ;TYPE 4015 SYNCHRONOUS COMMUNICATIONS CONTROLLER
 DUSR JRB=1
 DUSR BUTN=1
 DUSR RCU=1
 DUSR SERIO=0
 DUSR FGATE=0
 DUSR HDMA1=1
 DUSR HDMA2=1
        DUSR   KW7S=1            ;NUMBER OF KW7'S
```

```
;       EXECUTIVE FOR REAL TIME OPERATION OF ARMY MILITARIZED
;
;       AIRBORNE COMPUTER (AN/UYK-34) WHICH CONTROLS THE
;
;       GPS/DOPPLER SYSTEM AND IS ENTIRELY CORE-RESIDENT
;

        TITL RTOS

        ENT  IOX, FORK, QUIT, XMIT, RCV, WAIT, PTY, BRK
        ENT  ENQU, DEQU
        ENT     INTP
        IFN SMON
        ENT  BMON, EMON, MON
        ENDC

        ENT  SYS , CPTY, PMSK, QUE , RTC
        ENT  QSTK, QPNT, JSTK, JPNT
        ENT  CSTK, CPNT, CUCB, CLK , JOB , CST
        ENT  CHOT, DUCB, CHIN, BTAB, TTIO, TERM
        ENT  SERV, STAK, BCHR, TTYO, TTYI, TTYO
        ENT  DISN, CHRO, CHRI, PTAB, IOEND, SHED
        ENT  IOER, UNER, DCB1



        ZREL

        RDX 8

PAGE ZERO TRANSFER VECTORS

TRAN:   IO      ;ENTRY POINT FOR  IOX COMMAND
        FRK     ;ENTRY POINT FOR  FORK COMMAND
        SCHD-1          ;ENTRY POINT FOR  QUIT COMMAND
        XM      ;ENTRY POINT FOR  XMIT COMMAND
        RV      ;ENTRY POINT FOR  RCV COMMAND
        CLK     ;ENTRY POINT FOR  WAIT COMMAND
        PR      ;ENTRY POINT FOR  PTY COMMAND
        BR      ;ENTRY POINT FOR  BRK COMMAND
SERV    PRIOR   ;INTERRUPT PRIORITY CONTROLLER
STAK    IOSTK   ;JOB STACKER FOR I/O CALLS
        ENQ     ;ENTRY POINT FOR  ENQU COMMAND
        DEQ     ;ENTRY POINT FOR  DEQU COMMAND
```

```
;DEFINE SYSTEM INSTRUCTION CALLS

IOX=    JSR @ TRAN
FORK=   JSR @ TRAN+1
QUIT=   JSR @ TRAN+2
XMIT=   JSR @ TRAN+3
RCV=    JSR @ TRAN+4
WAIT=   JSR @ TRAN+5
PTY=    JSR @ TRAN+6
BRK=    JSR @ TRAN+7
ENQU=   JSR @ TRAN+12
DEQU=   JSR @ TRAN+13


;PAGE ZERO SYSTEM STORAGE AREA AND PARAMTERS

SYS:    0          ;MODE SWITCH: 0=>USER, 1=>SYSTEM
AC2:    0          ;AC2 STORAGE WHILE IN SYSTEM MODE
AC3:    0          ;AC3 STORAGE WHILE IN SYSTEM MODE
RTN:    0          ;RETURN ADDRESS STORAGE (SYSTEM MODE)
.SCHD:   SCHD      ;ENTRY POINT TO JOB SCHEDULER

         .RDX 8
         .IFN SMON
BMON:   BMON
EMON:   EMON
MON:    EMON
        .ENDC


;CONTROL BLOCK RELATIVE ENTRIES DEFINITIONS

;TASK CONTROL BLOCK (TCB)

.DUSR   TLINK=  0       ;LINK WORD
.DUSR   TCRY=   1       ;CARRY IN BIT 0
.DUSR   TPTY=   1       ;PRIORITY IN RIGHT 8 BITS
.DUSR   TAC0=   2       ;ACCUMULATOR STORAGE
.DUSR   TAC1=   3
.DUSR   TAC2=   4
.DUSR   TAC3=   5
.DUSR   TPC=    6       ;PROGRAM COUNTER
;INTERRUPTED MACHINE STATUS STORAGE BLOCK

.DUSR   INMSK=  -2      ;NEW MACHINE MASK
.DUSR   IOMSK=  -1      ;OLD    "     "
.DUSR   ICRY=   0       ;CARRY IN BIT 0
.DUSR   IAC0=   1       ;ACCUMULATOR STORAGE
.DUSR   IAC1=   2
.DUSR   IAC2=   3
.DUSR   IAC3=   4
.DUSR   IPC=    5       ;PROGRAM COUNTER
.DUSR   IDUCB=  6       ;DEVICE UNIT CONTROL BLOCK
```

```
;
;DEVICE UNIT CONTROL BLOCK
;
 DUSR    DTCBA=  0        ;TCB ADDRESS
 DUSR    DFLNK=  DTCBA    ;FORWARD LINKING
 DUSR    DTEMP=  1        ;TEMPORARY STORAGE
 DUSR    DGSR=   3        ;GET/STORE CHAR ROUTINE
 DUSR    DVCDE=  4        ;DEVICE CODE
 DUSR    DDADR=  5        ;DATA POINTER
 DUSR    DCNT=   6        ;DATA COUNT
 DUSR    DQBSY=  7        ;QUEUE BUSY INDICATOR
 DUSR    DCMDE=  10       ;CHAR. MODE (ASCII OR IMAGE)
 DUSR    DPRTY=  11       ;PARITY ROUTINE ADDR.
 DUSR    DNIOR=  12       ;NEXT I/O ROUTINE
 DUSR    DIDBO=  13       ;ADDR INTERRUPT DATA BLOCK (OUTPUT)
 DUSR    DTERM=  14       ;ADDR OF INPUT TERMINATORS
 DUSR    DIDBI=  15       ;ADDR INTERRUPT DATA BLOCK (INPUT)
 DUSR    DERTN=  16       ;ERROR RETURN ADDRESS
 DUSR    DMODE=  17       ;OPERATING MODE
 DUSR    DBRK=   20       ;BREAK ENABLED INDICATOR

;
;
;CALLING SEQUENCE CONTROL PARAMETERS
;
 DUSR    INMBR=  0        ;
 DUSR    ICNTL=  1        ;
 DUSR    IDPTR=  2        ;
 DUSR    IDCNT=  3        ;
 DUSR    IERTN=  4        ;
                          ;
 DUSR    FPTY=   0        ;
 DUSR    FTADR=  1        ;
                          ;
 DUSR    RCHAN=  0        ;
 DUSR    XCHAN=  0        ;
 DUSR    XRWRD=  0        ;
 DUSR    XRMES=  CHAN     ;
                          ;
 DUSR    BCODE=  0        ;
                          ;
 DUSR    PPTY=   0        ;
                          ;
 DUSR    WTIME=  0        ;
                          ;
;RTOS ERROR FLAGS
                          ;
 DUSR    DEVER=0          ;DEVICE NUMBER ERROR (.IOX)
 DUSR    UNER=-3          ;UNIT NUMBER ERROR (.IOX)
 DUSR    WCER=-2          ;WORD COUNT ERROR (.IOX)
```

```
;
;*****************************************************
;
;        ENTRY POINT FOR  RCV INSTRUCTION CALL
;FORMAT
;         RCV
;        ;CHAN    ;CHANNEL NUMBER
;         RETURN ;RETURN (WHEN MESSAGE RECEIVED)
;
;*****************************************************
;
         NREL
;
RV       ISZ  SYS          ;INDICATE SYSTEM MODE
         STA 2, AC2        ;SAVE AC2
         STA 3, RTN        ;SAVE AC3
         ISZ  RTN          ;INCREMENT FOR RETURN ADDR
         LDA 2,RCHAN,3     ;GET CHANNEL #
         LDA 3,CHAN.       ;GET # CHANNELS IN SYSTEM
         MOVL# 2,2,SNC     ;IS CHL. # > OR = ZERO?
         SUBZ# 3,2,SZC     ;YES, DOES REQUESTED CHANNEL EXIST?
         JMP  SCHD         ;NO, ERROR => TERMINATE JOB
         LDA 3,XRCON       ;GET BASE ADDR CHANNEL TABLE
         ADD 2,3                  ;HAVE TABLE ADDRESS
         STA 3,TEMP        ;SAVE IT
         LDA 2,XRWRD,3     ;GET TABLE WORD(CHANNEL STATUS)
         MOVL# 2 2 SZC     ;CHANNEL ACTIVE WITH A .XMIT ?
         JMP .+3                  ;YES, CONTINUE
         MOV 2 2 SZR       ;NO, CHANNEL ACTIVE WITH A .RCV?
         JMP  SCHD         ;YES,TERMINATE NEW JOB
         JSR @PQBLK        ;CREATE TCB FOR .RCV JOB
         LDA 0,@TEMP       ;GET TABLE WORD AGAIN
         MOVZL# 0,0,SNC    ; XMIT ARRIVED?
         JMP  RV2          ;NO
         LDA 3,TEMP        ;GET TABLE ADDRESS
         LDA 3,XRMES,3     ;GET 16 BIT MESSAGE
         STA 3,TAC3,2      ;PUT INTO AC3 STORAGE
         MOVZL 0,0,SNR     ;.XMIT HELD?
         JMP .RV1+1        ;NO
         JSR @PPSH         ;YES, ACTIVATE .RCV JOB
         MOVZR 0,2         ;GET ADDRESS .XMIT JOB TCB
RV1:     SUB 0,0                  ;CLEAR AC0
         STA 0,@TEMP       ;CLEAR CHANNEL WORD
         JMP  SHED         ;EXIT TO SCHEDULER(ACTIVATE JOB)
;
;
RV2:     STA 2,@TEMP       ;ENTER ADDR IN CHANNEL TABLE
         JMP .SCHD         ;EXIT TO SCHEDULER
;
CHAN.    CHAN              ;NUMBER OF .XMIT/.RCV CHANNELS
;
XRCON:    CST.             ;BASE ADDR CHANNEL STATUS TABLE
;
PQBLK:    QBLK             ;CREATE TCB SUBROUTINE ADDRESS
PPSH:    JSPSH             ;PUT TCB ADDR. IN JOB PENDING STACK
```

```
;*****************************************************
;
;       ENTRY POINT FOR  XMIT INSTRUCTION CALL
;FORMAT
;        XMIT             XMIT
;        CHAN             @CHAN          ;CHANNEL # (@ IMPLIES HOLD)
;        RETURN           RETURN
;
;*****************************************************
;
XM      ISZ  SYS         ;INDICATE SYSTEM MODE
        STA 2, AC2       ;SAVE AC2
        STA 3, RTN       ;SAVE AC3
        ISZ  RTN         ;INCREMENT FOR RETURN ADDR
        LDA 3,XCHAN,3    ;PICKUP PARAMETER WORD
        STA 3,TEMP+1     ;SAVE IT
        MOVL 3,3         ;IGNORE HOLD BIT
        MOVZR 3,3        ;BY SHIFTING IT OUT
        LDA 2,CHAN       ;GET # CHANNELS IN SYSTEM
        SUBZ# 2,3,SZC    ;DOES REQUESTED CHANNEL EXIST?
        JMP  SCHD        ;NO, ERROR => TERMINATE JOB
        LDA 2,XRCON      ;GET BASE ADDR CHANNEL TABLE
        ADD 2,3                  ;HAVE TABLE ADDR
        STA 3,TEMP       ;SAVE IT
        JSR @PQBLK       ;CREATE QUEUE ENTRY BLOCK
        LDA 3,@TEMP      ;PICKUP CHANNEL WORD
        MOVZL# 3,3,SZC   ;ANOTHER XMIT THERE?
        JMP  SHED        ;YES, QUICK EXIT
        MOV 3,3,SNR      ;NO, A  RCV WAITING?
        JMP  XM1         ;NO
        LDA 0,TAC0,2     ;YES, GET MESSAGE (IE AC0)
        STA 0,TAC3,3     ;STORE AS AC3 IN RCV BLOCK
        MOV 3,1                  ;SAVE ADDR RCV BLOCK IN AC1
        JSR JSPSH        ;ENTER INTO PENDING STACK
        MOV 1,2                  ;GET ADDR RCV BLOCK BACK IN AC2
        JMP  RV1         ;CLEAR CHANNEL WORD
                         ;EXIT TO SCHEDULER(ACTIVATE JOB)
;
XM1     SUB 1,1                  ;CLEAR AC1
        LDA 0,TEMP+1     ;PICKUP PARAMETER WORD
        MOVZL# 0,0,SZC   ;SUSPEND XMIT JOB?
        MOV 2,1                ;YES
        MOVZL 1,1        ;NO, SET BIT 0
        MOVOR 1,1
;
        LDA 3,TEMP       ;RESTORE AC3 TO CHANNEL TABLE
        STA 1,XRWRD,3    ;STORE IN CHANNEL TABLE
        LDA 1,TAC0,2     ;GET MESSAGE (IE AC0)
        STA 1,XRMES,3    ;SAVE IN MESSAGE TABLE
        MOVZL# 0,0,SZC   ;SUSPEND XMIT JOB?
        JMP  SCHD        ;YES,EXIT TO SCHEDULER
        JMP  SHED        ;NO, EXIT "    " (ACTIVATE JOB)
```

```
;**************************************************
;
;       ENTRY POINT FOR  PTY INSTRUCTION CALL
;FORMAT:
;        PTY
;       <PRIORITY>        ;NEW PRIORITY
;       <RETURN>
;
;**************************************************
;
PR      ISZ  SYS          ;INDICATE SYSTEM MODE
        STA 2, AC2        ;SAVE AC2
        INC 3,3                   ;INCREMENT RETURN ADDRESS
        STA 3, RTN        ;SAVE IT
        LDA 2,PPTY-1,3    ;GET NEW PRIORITY
        JMP  FRK1         ;GO HANDLE
;
;**************************************************
;
;       ENTRY POINT FOR  FORK SYSTEM CALL
;FORMAT:
;        FORK
;       <PRIORITY>        ;NEW TASK PRIORITY (0=NO CHANGE)
;       <ADDRESS>         ;ADDRESS NEW TASK
;       <RETURN>
;
;**************************************************
;
FRK:    ISZ  SYS          ;INDICATE SYSTEM MODE
        STA 2, AC2        ;SAVE AC2
        LDA 2, CPTY       ;PICKUP CURRENT TASK PRIORITY
        STA 2,TEMP        ;SAVE IT
        LDA 2,FPTY,3      ;GET NEW TASK PRIORITY
        MOV 2,2,SZR       ;NEW PRIORITY ZERO?
        STA 2, CPTY       ;NO, USE IT
        INC 3,3                   ;INCREMENT AC3
        LDA 2,FTADR-1,3           ;GET NEW TASK ADDRESS
        STA 2, RTN        ;SET AS RETURN ADDRESS
        INC 3,3                   ;INCREMENT AC3
        STA 3,TEMP+1      ;SAVE IT (OLD TASK RETURN ADDRESS)
        JSR  QBLK         ;CREATE QUEUE BLOCK
        JSR JSPSH         ;PUT IN JOB PENDING STACK
        LDA 2,TEMP+1      ;GET OLD TASK RETURN ADDRESS
        STA 2, RTN        ;SET AS RETURN
        LDA 2,TEMP        ;PICKUP OLD TASK PRIORITY
FRK1    STA 2, CPTY       ;SET IT
        JSR  QBLK         ;CREATE QUEUE BLOCK
        JMP  SHED         ;EXIT TO SCHEDULER(ACTIVATE JOB)
;
;
TEMP:     BLK 2           ;TEMPORARY STORAGE USED BY SYSTEM
                          ;META-INSTRUCTION SERVICING PROGRAMS
```

```
;****************************************************

;     JOB SCHEDULER FOR THE REAL-TIME OPERATING SYSTEM

;****************************************************

        ISZ  SYS          ;INDICATE SYSTEM MODE
SCHD:   INTDS             ;DISABLE INTERRUPT MOMENTARILY
        LDA 2,@ JPNT      ;CHECK JOB PENDING STACK
        MOV 2,2,SNR       ;STACK EMPTY?
        JMP  SHD3         ;YES  ACTIVATE JOB
        INTEN             ;NO  ENABLE INTERRUPT
        DSZ  JPNT         ;DECREMENT POINTER

;****************************************************
;
;       ENTRY POINT TO SCHEDULER WITH TCB ADDRESS TO BE
;       ACTIVATED ALREADY IN AC2
;
;****************************************************

SHED:   STA 2,TEMP        ;SAVE ADDRESS
        LDA 0,TPTY,2      ;GET C(ENTRY+1)
        MOVZL 0,0         ;C(AC0)=2*PRIORITY
        LDA 3,SHCON       ;GET ADDR POINT. TO TOP OF QUEUE

SHD1:   LDA 2,0,3         ;GET NEXT ENTRY ADDRESS
        MOV 2,2,SNR       ;END OF QUEUE?
        JMP .SHD2         ;YES. MAKE ENTRY
        LDA 1,TPTY,2      ;NO. GET C(ENTRY+1)
        MOVZL 1,1         ;C(AC1)=2*PRIORITY
        ADC 0,1
        MOVZL 1,1,SNC     ;FOUND SPOT?
        JMP .SHD2         ;YES. MAKE ENTRY
        MOV 2,3                   ;NO. UPDATE QUEUE POINTER
        JMP .SHD1         ;CONTINUE

SHD2:   LDA 1,TEMP        ;PICKUP ADDR NEW QUEUE ENTRY
        STA 2,@TEMP       ;SET FWD LINK
        STA 1,TLINK,3     ;FWD LINK OF PREVIOUS RESET
        JMP  SCHD         ;GO BACK FOR ANY MORE
```

```
;
;
;****************************************************
;
;       ACTIVATE JOB OF HIGHEST PRIORITY
;
;****************************************************
;
SHD3    LDA 2,@SHCON       ;GET ADDR. TOP ENTRY IN QUEUE
        MOVZ 2,2,SNR       ;QUEUE EMPTY?
        JMP  SHD4          ;YES. SET UP NULL JOB
;****************************************************
;
;       MONITOR WHO HAS CONTROL OF THE SYSTEM
;
;****************************************************
         IFN SMON                                      ;
        SUB 3,3                                        ;
        STA 3,@.MON                                    ;
        DSZ .MON                                       ;
        LDA 3,TPC,2                                    ;
        STA 3,@.MON                                    ;
        DSZ  MON                                       ;
        ADC 0,0                                        ;
        STA 0,@.MON                                    ;
        LDA 0,.MON                                     ;
        LDA 1,.BMON                                    ;
        SUBZ# 1,0,SNC                                  ;
        LDA 0,.EMON                                    ;
        STA 0,.MON                                     ;
         ENDC                                          ;
        LDA 0,TLINK,2      ;NO. GET FWD LINK
        STA 0,@SHCON       ;RESET POINTER
        LDA 0,TPTY,2       ;PICKUP PRIORITY & CARRY
        MOVZL 0,0          ;IGNORE CARRY BIT FOR NOW
        MOVZR 0,0                                      ;
        STA 0,.CPTY        ;SET CURRENT PRIORITY
        LDA 0,TAC0,2       ;RESTORE AC0
        LDA 1,TAC1,2       ;RESORE AC1
        LDA 3 TPC 2        ;PICKUP PROGRAM COUNTER
        STA 3 TEMP         ;SAVE IT
        LDA 3 TPTY 2       ;GET CARRY
        MOVL 3 3           ;RESTORE IT
        SUBC 3 3           ;CLEAR AC3
        STA 3 .SYS.        ;RESET TO USER MODE
        ISZ .QPNT          ;PUSH TCB ADDRESS
        STA 2 @.QPNT       ;BACK ON QUEUE STACK
        LDA 3 TAC3 2       ;RESTORE AC3
        LDA 2 TAC2 2       ;RESTORE AC2
        INTEN              ;RE-ENABLE THE INTERRUPT
        JMP @TEMP          ;EXIT TO JOB
                                                       ;
```

```
;*****************************************************
;
;       ACTIVATE A NULL JOB
;
;*****************************************************

SH04    STA 2,0                 ;SET LOCATION 0 = 0
        STA 2,SYS       ;RESET TO USER MODE
        INTEN           ;TURN INTERRUPT ON
        JMP 0           ;JUMP TO THE NULL TASK
                        ;TASK =   07  JMP 0
SHCON    QUE            ;ADDRESS VECTOR


;*******************************************************

        ROUTINE TO POP ADDRESS FROM JOB QUEUE STACK
;
;CALLING SEQUENCE:
;       JSR QSPOP
;       <RETURN>                ;TCB ADDRESS IN AC2
;
;*******************************************************

QSPOP:  INTDS           ;DISABLE INTERRUPT MOMENTARILY
        LDA 2,@.QPNT    ;PICKUP TCB ADDRESS FROM STACK
        MOV 2,2,SZR     ;F ADDR. =0 => NO MORE TCB'S AVAILABLE


        JMP  .+3                ;LEGAL TCB ADDRESS FOUND
        .IFE SHALT
        HALT            ;NO TCB'S AVAILABLE, ERROR HALT
        JMP  .-1
        .ENDC
         IFN SHALT      ;NO TCB'S AVAILABLE ON STACK
         EXTN .SHLT     ;MUST BRANCH TO USER
        JMP @.+1        ;SUPPLIED ROUTINE TO
         SHLT           ;HANDLE END OF TCB TABLES
         ENDC

        DSZ .QPNT       ;NO, DECREMENT STACK POINTER
        INTEN           ;RE-ENABLE THE INTERRUPT
        JMP 0,3                 ;RETURN

 QPNT   0               ;STACK POINTER
```

```
;*****************************************************
;
;        ROUTINE TO PUSH ADDRESS INTO JOB PENDING STACK
;N.B.  NO OVERFLOW CHECK REQUIRED
;
;CALLING SEQUENCE:
;        LDA 2,<ADDR>    ;ADDR IN AC2
;        JSR JSPSH
;        <RETURN>
;
;JSPSH IS RE-ENTRANT
;
;*****************************************************

JSPSH:   INTDS              ;DISABLE INTERRUPT MOMENTARILY
         ISZ .JPNT          ;INCREMENT POINTER
         INTEN              ;RE-ENABLE INTERRUPT
         STA 2,@.JPNT       ;STORE ADDRESS
         JMP 0,3                    ;EXIT

 JPNT:   0                  ;JOB PENDING STACK POINTER




;*****************************************************
;
;        ROUTINE TO PUSH ADDRESS INTO QUEUE STACK
;
;CALLING SEQUENCE
;        (IDENTICAL TO JSPSH)
;
;*****************************************************

QSPSH:   INTDS              ;DISABLE INTERRUPT MOMENTARILY
         ISZ .QPNT          ;INCREMENT POINTER
         INTEN              ;RE-ENABLE INTERRUPT
         STA 2,@.QPNT       ;STORE ADDRESS
         JMP 0,3                    ;EXIT


;. IOX WORD COUNT ERROR

 WCER:   JSR .IOER
         WCER               ;WORD COUNT WAS NEGATIVE
```

```
; IOX DEVICE NUMBER ERROR RETURN

 DVER   JSR  IOER
        DEVER           ;DEVICE NUMBER ERROR CODE

; IOX UNIT ERROR RETURN

 UNER   JSR  IOER
        UNER            ;UNIT NUMBER ERROR

;GENERAL  IOX PARAMETER ERROR ROUTINE
;CALLING SEQUENCE
        JSR  IOER
        (ERROR CODE)

 IOER   LDA 0, @3       ;PICKUP ERROR CODE
        LDA 3,TAC3,2    ;RESTORE AC3 (IE ADDR .IOX+1)
        STA 0,TAC3,2    ;SAVE AS RETURNED AC3
        LDA 0,IERTN,3   ;GET ERROR RETURN ADDR
        STA 0,TPC,2     ;ENTER AS RETURN PC
        JSR JSPSH       ;ACTIVATE JOB
         QUIT           ;EXIT TO SCHEDULER

C5      5               ;+5
IOCON   HANTB           ;ADDRESS DEVICE HANDLER TABLE

 DEV    HANTE-HANTB+1   ; = # OF DEVICES IN SYSTEM

 CPTY   0               ;PRIORITY OF JOB NOW IN PROGRESS




;*****************************************************
;
;A GENERAL NON REENTRANT ROUTINE TO CREATE A QUEUE BLOCK
;
;USED ONLY BY SYSTEM CALLS (NO REENTRANCY REQUIRED)
;
;CALLING SEQUENCE
;       (AC2,AC3,RETURN ADDR STORED IN .AC2,.AC3,.RTN. )
;       JSR  @BLK
;       <RETURN>        ;(AC2)=ADDR OF CREATED QUEUE BLK
;
;PRIORITY SET BY C(CPTY)
;
;*****************************************************
```

```
[illegible]    STA 3, ORTN        ;SAVE RETURN ADDRESS
               JSR @.FQR          ;GET QUEUE BLOCK
               STA 0,TAC0,2       ;ENTER AC0 IN QUEUE BLOCK
               STA 1,TAC1,2       ;ENTER AC1 IN QUEUE BLOCK
               MONITOR ALL MAIN SYSTEM CALLS
               FIRST WORD IS ADDRESS OF TCB REQUESTOR
               SECOND WORD DEFINES WHO IT WILL GO TO
               .IFN .MON
               LDA 3, [illegible]
               STA 3,@ MON
               DSZ  MON
               LDA 3, RTN
               STA 3,@ MON
               DSZ  MON
               ADC 0,0
               STA 0,@ MON
               LDA 0, MON
               LDA 1, EMON
               SUBZ# 1,0,SNC
               LDA 0, EMON
               STA 0, MON
               LDA 1,TAC1,2
                ENDC
               SUB 0 0                    ;ZERO FORWARD LINK OF TCB
               STA 0,TLINK,2      ;BEING CREATED
               LDA 0, CPTY        ;PICKUP CURRENT PRIORITY
               LDA 3,P377         ;GET PRIORITY MASK
               AND 3,0                    ;PRIORITY IN CORRECT RANGE
               ADDR 0,0           ;SAVE CARRY IN PRIORITY WORD
               STA 0,TPTY,2       ;ENTER IT IN QUEUE BLOCK
               LDA 0, AC2         ;GET AC2
               STA 0,TAC2,2       ;ENTER IN QUEUE BLOCK
               LDA 0, AC3         ;GET AC3
               STA 0,TAC3,2       ;ENTER IN QUEUE BLOCK
               LDA 0, RTN         ;GET RETURN ADDRESS
               STA 0,TPC,2        ;ENTER IN QUEUE BLOCK
               LDA 0,TAC0,2       ;RESTORE ORIGINAL AC0
               JMP @ ORTN         ;EXIT

ORTN.          0                  ;SUBROUTINE RETURN ADDRESS
P377           377                ;8 BIT MASK
```

```
;*****************************************************
;
;       ENTRY POINT FOR  IOX INSTRUCTION CALL
;FORMAT
;         IOX
;        [DEV]                ;DEVICE NUMBER          (INMBR)
;        [CONTROL]            ;DEVICE CONTROL WORD    (ICNTL)
;        [POINTER]            ;DATA POINTER           (IDPTR)
;        [COUNT]                      ;DATA COUNT             (IDCNT)
;        [ERROR]                      ;ERROR RETURN           (IERTN)
;        [RETURN]             NORAML RETURN
;
;*****************************************************

IO      ISZ   SYS.            ;INDICATE SYSTEM MODE
        STA 2, AC2            ;SAVE AC2
        LDA 2,C5              ;GET CONSTANT +5
       ADD 3,2                        ;CALCULATE NORMAL RETURN ADDR
        STA 2, RTN            ;ENTER RETURN PC
        STA 3, AC3            ;SAVE AC3
        JSR  QBLK             ;CREATE QUEUE BLOCK
        LDA 3, AC3            ;RESTORE AC3
        LDA 1,INMBR,3         ;GET DEVICE NUMBER
        LDA 0,IDCNT,3         ;GET DATA COUNT
        MOVL 0,0,SZC          ;COUNT >=0?
        JMP  WCER             ;NEGATIVE => WORD COUNT ERROR
        LDA 0,ICNTL,3         ;GET DEVICE CONTROL WORD
        LDA 3, DEV            ;GET NUMBER OF DEVICES IN SYSTEM
        SUBZ# 3,1,SZC         ;A LEGAL DEVICE NUMBER?
        JMP  DVER             ;NO, ERROR => TERMINATE JOB
        LDA 3,IOCON           ;GET TOP ADDRESS DEVICE TABLE
        ADD 1,3                       ;HAVE TRANSFER ADDRESS POINTER
        SUB 1,1                       ;CLEAR AC1
        STA 1,TLINK,2         ;CLEAR FWD LINK OF QUEUE ENTRY
        LDA 3,0,3             ;TRANSFER TO DEVICE
        JMP @-1,2             ;  INITIALIZATION ROUTINE
```

```
; ****************************************************************
;
;        ENTRY POINT FOR  BRK INSTRUCTION CALL
;FORMAT
;         BRK
;         BCODE                ;8 BIT CODE OF BREAK CHARACTER
;         RETURN               ;RETURN HERE IF BREAK OCCURS
;
; ****************************************************************

BR       ISZ  SYS          ;INDICATE SYSTEM MODE
         STA 2, AC2        ;SAVE AC2
         STA 3, RTN        ;SAVE AC3
         ISZ  RTN          ;INCREMENT FOR RETURN ADDRESS
         LDA 2, BCODE, 3   ;GET NEW BREAK CHARACTER
         LDA 3, BCHR       ;GET OLD BREAK CHAR.
         STA 2, BCHR       ;SAVE NEW BREAK CHAR.
         MOVL 3, 3, SZC    ;WAS BREAK REQUEST ACTIVE?
         JMP  BR1          ;NO, SETUP NEW REQUEST
         MOVL 2, 2, SZC    ;YES, CHECK IF NEW BREAK CHAR -VE
         ADCZL 3, 3, SKP   ;YES, GENERATE -2
         ADC 3, 3                  ;NO, GENERATE -1
         LDA 2, BJOB       ;GET OLD BREAK JOB TCB
         STA 3, TAC3, 2    ;SET RETURN AC3
         JSR JSPSH         ;ACTIVATE OLD BREAK JOB
BR1      JSR  QBLK         ;CREATE QUEUE BLOCK FOR . BRK JOB
         LDA 3, BCHR       ;GET NEW BREAK CHARACTER
         MOVL 3, 3, SZC    ;A NEW BREAK REQUEST?
         JMP  .BR2         ;NO, IT WAS A TERMINATE REQUEST
         STA 2, .BJOB      ;YES, SAVE . BRK TCB ADDR.
         JMP @@SCHD        ;EXIT TO SCHEDULER

BR2      LDA 3, BR3        ;SET AC3 = -3
         STA 3, TAC3, 2    ;SET RETURN AC3
         JMP @@SHED        ;SCHEDULE TASK IMMEDIATELY

BR3      -3
```

```
;***************************************************************
;
;ENTRY POINT FOR [illegible] CALL
;FORMAT
        WAIT
        COUNT                         ;CLOCK INTERVAL COUNT
        RETURN              ;NORMAL RETURN
;
;***************************************************************

CLK     ISZ  SYS          ;INDICATE SYSTEM MODE
        STA 2, AC2        ;SAVE AC2
        INC 3,3                   ;INCREMENT AC3 FOR RETURN
        STA 3, RTN        ;SAVE IT
        LDA 3,WTIME-1,3           ;PICKUP COUNT PARAMETER
        STA 3,CVAR+1      ;SAVE IT
        JSR  QBLK         ;CREATE QUEUE BLOCK ENTRY
        LDA 0 CVAR+1      ;PICKUP COUNT
        MOV 0 0 SNR       ;IS IT = ZERO ?
        JMP @QSHED        ;YES, GO AND RESCHEDULE
        INTDS             ;DISABLE INTERRUPT MOMENTARILY
        LDA 3,@ CPNT      ;PICKUP CLOCK QUEUE BLOCK
        INTEN             ;RE-ENABLE INTERRUPT
        DSZ  CPNT         ;DECREMENT POINTER



        STA 2,2,3         ;ENTER JOB QUEUE BLOCK ADDR
        STA 3,CVAR        ;SAVE ADDR NEW QUEUE BLOCK
        ISZ  CUCB+3       ;SET CLOCK QUEUE BUSY SWITCH
        LDA 3,CQUEU       ;GET ADDR POINT  TO TOP OF QUEUE

;FIND PLACE IN CLOCK QUEUE

CLK1    LDA 2,0,3         ;GET NEXT ENTRY ADDR
        MOV 2,2,SNR       ;END OF QUEUE?
        JMP  CLK2+3       ;YES, MAKE ENTRY
        LDA 1,1,2         ;NO, GET COUNT OF THIS ENTRY
        SUB 1,0                   ;SUBTRACT FROM REMAINING TOTAL
        MOVZL# 0,0,SZC    ;FOUND SPOT?
        JMP  CLK2         ;YES, MAKE ENTRY
        MOV 2,3                   ;NO, UPDATE AND CONTINUE
        JMP  CLK1
```

```
.MAKE SUITABLE ENTRY IN CLOCK QUEUE

                                 ;RESTORE COUNT TO CORRECT VALUE
CLI3:   ADD 1,0                  ;UPDATE COUNT FOR NEXT ENTRY
        SUB 0,1
        STA 1,1,2       ;NEXT ENTRY'S COUNT RESET
        LDA 1,CVAR      ;PICKUP ADDR NEW QUEUE ENTRY
        STA 2,@CVAR     ;SET FWD LINK
        STA 1,0,3       ;RESET FWD LNK OF PREVIOUS ENTRY
                                 ;GET NEW QUEUE ENTRY ADDR IN AC3
        MOV 1,3
        STA 0,1,3       ;SET ITS COUNT INTERVAL
                                 ;CLEAR AC0
        SUB 0,0
        STA 0, CUCB+3   ;INDICATE CLOCK QUEUE AVAILABLE
        LDA 0, CUCB+2   ;PICKUP CLOCK FREQUENCY
        LDA 1, CUCB     ;PICKUP CLOCK ACTIVE SWITCH
        MOV 1,1,SZR     ;CLOCK ACTIVE?
        JMP @QSCHD      ;YES, EXIT TO SCHEDULER
        DOAS 0,RTC      ;NO, START IT UP
        ISZ CUCB        ;INDICATE CLOCK ACTIVE
        JMP @QSCHD      ;EXIT TO SCHEDULER
CVAR     BLK 2          ;TEMPORARY STORAGE
QSHED    SHED           ;ADDRESS TO RESCHEDULE
CQUEU    RTC            ;ADDR POINTER TO CLOCK QUEUE


;*************************************************
;
;       REAL-TIME CLOCK INTERRUPT SERVICE ROUTINE
;
;*************************************************

CKSER:   JSR PRIOR      ;RE-ARRANGE PRIORITIES
         37             ;NEW INTERRUPT MASK
         0              ;OLD MASK STORAGE
         0              ;CARRY
         0              ;AC0
         0              ;AC1
         0              ;AC2
         0              ;AC3
         0              ;PC
 RTC     0              ;FIRST ENTRY CLOCK QUEUE POINTER
         NIOS RTC       ;STARTUP CLOCK AGAIN
         ISZ CUCB+1     ;INCREMENT OVERCOUNT WORD
         LDA 3, CUCB+3  ;FETCH QUEUE AVAILABILITY SWITCH
         MOV 3,3,SZR    ;QUEUE AVAILABLE?
         JMP CKS1       ;NO, GO INTO OVERCOUNT
         JSR CKS3       ;DECR INTERVAL CNT (AC2=C(RTC))
         JMP CKS4       ;GO ACTIVATE JOB
```

```
CKS1    LDA 3,CKSER+2       ;RECALL ORIGINAL PRIORITY WORD
        DOBC 3,CPU          ;RESTORE OLD MASK, INTDS
        SKPDZ RTC           ;RTC INTERRUPTED AGAIN?
        JMP  CKS2           ;YES,SERVICE NEW REQUEST
        LDA 2,INCLK         ;NO, DISMISS INTERRUPT
        JMP @.+1            ;BRANCH TO DISMISSAL PROGRAM
        DISN+4                          ;RESTORE AC2,AC3,CARRY,PC

CKS2    LDA 3,CKSER+1       ;GET NEW INTERRUPT MASK
        DOBS 3,CPU          ;RESTORE MASK  HARDWARE,INTEN
        JMP  RTC +1         ;SERVICE NEW RTC INTERRUPT

CKS3    DSZ 1,2                         ;DECR INTERVAL CNTR
        JMP  .+2                        ;NOT ZERO CHECK FOR OVERCOUNT
        JMP 0,3                         ;ZERO, GO ACTIVATE JOB
        DSZ  CUCB+1         ;DECREMENT OVERCOUNT
        JMP  .-4                        ;KEEP GOING AS OFTEN AS REQ'D
        JMP 1,3                         ;NO JOBS TO BE ACTIVATED


;REAL TIME CLOCK DEVICE CONTROL BLOCK

CUCB    0                   ;0=>CLOCK INACTIVE, NON 0=>ACTIVE
        0                   ;0=>NORMAL, OTHERWISE OVERCOUNT
        FREQ&3              ;CLOCK FREQUENCY (0,1,2,3)
        0                   ;0=>QUEUE AVAILABLE, 1=>QUEUE BUSY

CPNT    0        ;CLOCK STAC POINTER

CKS4    MOV 2,3                         ;GET ADDR TOP ENTRY IN QUEUE
        LDA 0,0,3           ;GET LINK TO NEXT ENTRY
        LDA 2,2,3           ;GET SUSPENDED TASK'S TCB ADDR
        INTDS               ;DISABLE INTERRUPT
        ISZ  CPNT           ;INCREMENT POINTER
        INTEN               ;RE-ENABLE THE INTERRUPT
        STA 3,@ CPNT        ;RETURN ADDRESS TO STACK
        JSR @QPSH           ;ACTIVATE JOB
        STA 0, RTC          ;RESET POINTER TO NEXT ENTRY
        MOV 0,2,SNR         ;IS QUEUE NOW EMPTY?
        JMP  CKS5           ;YES
        LDA 1,1,2           ;GET INTERVAL COUNT FOR NEXT ENTRY
        MOV 1,1,SZR         ;NO,IS NEXT INTERVAL COUNT ZERO?
        JSR  CKS3+3         ;NO,CHECK OVERCOUNT
        JMP  CKS4           ;CAN ACTIVATE JOB RIGHT NOW
        LDA 2,INCLK         ;NO,GET ADDR INTERRUPT DATA BLOCK
        JMP @PEND1          ;GO HANDLE AS GENERAL I/O END

CKS5    NIOC RTC            ;TURN OFF CLOCK
        STA 0, CUCB         ;MAKE CLOCK INACTIVE
        STA 0, CUCB+1       ;ZERO THE OVERCOUNT VALUE
        JMP  CKS5-2         ;EXIT

BCHR    0        ;BREAK HAR CODE, -VE => INACTIVE
BJOB    0        ;ADDR  TCB FOR SUSPENDED BREAK TASK
INCLK   CKSER+3                         ;INTERRUPT DATA BLOCK ADDR
PEND1    END1               ;I/O END INTERRUPT EXIT
QPSH    JSPSH                ;JOB PENDING STACK HANDLER
```

```
;*************************************************
;
;       INTERRUPT PROCESSOR
;
;       DETERMINE THE INTERRUPTING DEVICE AND
;       DISPATCH TO SERVICE ROUTINE
;
;NOTE   INTERRUPT SERVICING CAN BE SPEEDED UP BY
;       PLACING THE DISPATCH TABLE ON PAGE ZERO
;
;*************************************************

INTP:   STA 2,SAC2          ;SAVE AC2
        STA 3,SAC3          ;SAVE AC3

;POWER FAILURE INTERRUPT CHECKING

         IFN PWRFL
        SKPDZ CPU           ;WAS INTERRUPT A POWER FAIL?
        JMP @ DISP+1        ;YES BRANCH TO DEVICE HANDLER
         ENDC

        INTA 2              ;GET INTERRUPT DEVICE CODE
        MOV 2,3
        ANFNW 3
        000077
        ADFNW 3             ;GENERATE HANDLER ADDR POINTER
        DISP
        JMP @0,3            ;GO TO IT!!

;UNDEFINED DEVICE CAUSED THE INTERRUPT

NODEV   LDA 3, NIOC         ;GET CLEAR DEVICE COMMAND
        ADD 2,3                     ;FORM CLEAR INTERRUPT DEVICE
        STA 3, +1           ;INSTRUCTION,PLACE IN NEXT CORE LOC
        NIOC 0              ;INSTRUCTION STORAGE
DUNG    LDA 2,SAC2          ;RESTORE AC2
        LDA 3,SAC3          ;RESTORE AC3
        INTEN               ;ENABLE INTERRUPT
        JMP @0              ;RETURN TO INTERRUPTED PROGRAM

;INTERRUPT AC2,AC3 STORAGE

SAC2:   0
SAC3:   0
 NIOC   NIOC 0              ;CLEAR DEVICE INSTRUCTION

 DISP:  DISP                ;ADDR. INTERRUPT DISPATCH TABLE

;POWER FAILURE INTERRUPT HANDLER ADDRESS

         IFN PWRFL
         EXTN  PWR
         PWR
         ENDC
```

```
;*****************************************************

;       ROUTINE TO RE-ARRANGE PRIORITIES
;       AND SAVE MACHINE STATE
; CALLING SEQUENCE
;       JSR PRIOR           ;AC2,AC3 SAVED IN SAC2,SAC3
;       NMASK               ;NEW PRIORITY WORD
;       0                   ;OLD PRIORITY WORD STORAGE
;       0                   ;CARRY STORAGE
;       0                   ;AC0 STORAGE
;       0                   ;AC1 STORAGE
;       0                   ;AC2 STORAGE
;       0                   ;AC3 STORAGE
;       0                   ;PC STORAGE
;       DUCB                ;ADDR  OF DEVICE CONTROL BLOCK
;       (RETURN)            ;INTERRUPT ENABLED
;
;*****************************************************

PRIOR   LDA 2,SAC2          ;GET AC2
        STA 2,IAC2+2,3      ;SAVE IN CALLING ROUTINE
        LDA 2,SAC3          ;GET AC3
        STA 2,IAC3+2,3      ;SAVE IN CALLING ROUTINE
        LDA 2, PMSK         ;FETCH CURRENT PRIORITY WORD
        STA 2,IOMSK+2,3             ;SAVE IN CALLING ROUTINE
        LDA 2,INMSK+2,3             ;GET NEW PRIORITY
        STA 2, PMSK         ;ESTABLISH IT -- SOFTWARE
        DOBS 2,CPU          ;ESTABLISH IT -- HARDWARE
        LDA 2,0                     ;FETCH INTERRUPT PC
        STA 2,IPC+2,3       ;SAVE IN CALLING ROUTINE
        STA 0,IAC0+2,3      ;SAVE AC0 IN CALLING ROUTINE
        STA 1,IAC1+2,3      ;SAVE AC1 IN CALLING SEQUENCE
        SUBCR 2,2           ;GET CARRY
        STA 2,ICRY+2,3      ;SAVE IN CALLING ROUTINE
        LDA 2,IDUCB+2,3             ;GET DEVICE CONTROL BLOCK ADDR
        JMP IDUCB+3,3       ;RETURN (INTERRUPT ENABLED)

PMSK:   0                   ;CURRENT TASK PRIORITY STORAGE
```

```
;*****************************************************
;
;        TTY UNIT 0 INTERRUPT SERVICE ROUTINES
;
;*****************************************************
;
;INPUT
;
        TTYIO           ;ADDR OF DEVICE INIT. ROUTINE
TTIO:   JSR PRIOR       ;REARRANGE PRIORITIES
        3               ;NEW MASK
        0               ;OLD MASK STORAGE
        0               ;CARRY
        0               ;AC0
        0               ;AC1
        0               ;AC2
        0               ;AC3
        0               ;PC
         TTYO           ;ADDR TTY UNIT 0 CONTROL BLOCK
        DIAC 0,TTI      ;GET CHARACTER
        JMP .TTYI       ;GO PROCESS TTY INPUT

;
;OUTPUT
;
TTOO:   JSR PRIOR       ;REARRANGE PRIORITIES
        1               ;NEW MASK
        0               ;OLD MASK STO-AGE
        0               ;CARRY

        0               ;AC0
        0               ;AC1
        0               ;AC2
        0               ;AC3
        0               ;PC
        .TTYO           ;ADDR TTY UNIT 0 CONTROL BLOCK
        NIOC TTO        ;CLEAR TTY OUTPUT FLAGS
        JMP .TTYO       ;GO PROCESS TTY OUTPUT
```

```
**************************************************

        ENTER HT [illegible] WHEN BREAK CH[illegible] [illegible]ENCOUNTERED
        [illegible] TTY CONTROL [illegible]
        [illegible] ALREADY [illegible]

**************************************************

BREAK   MOV 2,0                      ;SAVE ADDR DUCB
        LDA 1,DTCBA,2       ;GET SUSPENDED TCB ADDR
        MOV 1,2,SNR         ;AN I/O OPERATION ACTIVE?
        JMP BRK3            ;NO, COMPLETE BREAK REQUEST HANDLING

BRK1    LDA 1,DFLNK,2       ;GET LINK TO NEXT STACKED BLOCK
        MOV 1,2,SNR         ;REACHED END OF STACKED I/O?
        JMP BRK2            ;YES, SETUP QUIT RETURN
        JSR @TPSH           ;NO, RETURN BLOCK TO QUEUE STACK
        JMP BRK1            ;CONTINUE

BRK2:   MOV 0,2                      ;RESTORE ADDR DUCB TO AC2
        STA 1,@DFLNK,2      ;SET LINK TO NEXT BLOCK=0

BRK3:   MOV 0,3                      ;GET DUCB ADDR IN AC3
        LDA 2,.BJOB         ;GET ADDR DORMANT QUEUE BLOCK
        LDA 3,DBRK,3        ;GET TTY UNIT #
        STA 3,TAC3,2        ;ENTER AS RETURN AC3
        JSR @QPSH           ;ACTIVATE BREAK JOB
        LDA 3,BRK3          ;PICK UP A -VE #
        STA 3,.BCHR         ;SET BREAK REQUEST INACTIVE
        MOV 0,3                      ;RESTORE DUCB ADDR TO AC3
        STA 1,.BJOB         ;SET BREAK REQUEST INACTIVE
        LDA 1,DMODE,3       ;GET MODE SWITCH
        LDA 2,DTCBA,3       ;GET SUSPENDED TCB ADDR
        MOV 2,2,SNR         ;AN I/O OPERATION IN PROGRESS?
        JMP BRK4-3          ;NO, DISMISS INTERRUPT
        MOVZR 1,1,SZR       ;YES, TEST MODE SWITCH
        JMP BRK4            ;OUTPUT MODE
        STA 1,DTCBA,3       ;INPUT MODE,SET DEVICE NOT BUSY
        JSR @TPSH           ;RETURN BLOCK TO QUEUE STACK
        MOV 0,2                      ;RESTORE DUCB ADDR TO AC2
        LDA 2,DIDBI,2       ;GET INTERRUPT DATA BLOCK ADDR
        JMP @PEND1          ;DISMISS INPUT INTERRUPT

BRK4    MOV 0,2                      ;RESTORE DUCB ADDR TO AC2
        LDA 1,BRK3          ;SET AC1 NEGATIVE
        LDA 0,QUIT          ;GET ".QUIT" SERVICING ADDR
        JMP HLTOT+5         ;OUTPUT MODE MUST BE TERMINATED

QUIT=    TRAN+2                      ;.QUIT SERVICING ADDR
TPSH    .SPSH               ;PUT ADDR. BACK ON STACK
```

```
;***************************************************************
;
;        GENERAL ROUTINE TO PRE-PROCESS ALL TTY INPUT
;
;     C(AC2)=ADDR DEVICE UNIT CONTROL BLOCK
;     C(AC0)=CHARACTER
;     AC0,AC2,AC3 ALREADY SAVED
;
;***************************************************************

TTYI    LDA 3,DBRK,2       ;GET BREAK ENABLE WORD
        COM# 3,3,SNR       ;BREAK ENABLED ON THIS TTY?
        JMP NOBRK          ;NO, GO ON
        LDA 3, BCHR        ;YES, GET CODE FOR BREAK CHAR
        SUB# 0,3,SNR       ;BREAK REQUEST?
        JMP BREAK          ;YES, GO SERVICE
NOBRK   LDA 3,DTCBA,2      ;GET TTY ACTIVE WORD
        MOV 3,3,SNR        ;IS TTY I/O IN PROGRESS?
        JMP DISN           ;NO - DISMISS INTERRUPT
        LDA 3,DMODE,2      ;GET TTY MODE WORD
        MOVZR# 3,3,SZR     ;INPUT? OR OUTPUT?
        JMP HLTOT          ;OUTPUT -- I/O ERROR
        MOVZR# 3,3,SZC     ;SUPRESS ECHO
        JMP CHIN           ;YES, GO HANDLE INPUT
        LDA 3, DOA         ;NO, PICKUP IOT CONSTANT
        STA 0,DTEMP,2      ;SAVE CHARACTER
        LDA 0,DVCDE,2      ;GET DEVICE CODE
        ADD 0,3                    ;ADD TO INSTRUCTION
        LDA 0,DTEMP,2      ;RESTORE CHARACTER
        STA 3,DTEMP,2      ;STORE INSTRUCTION
        JSR DTEMP,2        ;EXECUTE IT
        JMP  CHIN          ;GO HANDLE INPUT

DOA:    DOAS 0,1           ;INSTRUCTION


;*******************************************************************
;
;        GENERAL ROUTINE TO PRE-PROCESS ALL TTY OUTPUT
;
;     C(AC2)=ADDR DEVICE UNIT CONTROL BLOCK
;     AC2,AC3 ALREADY SAVED
;
;***************************************************************

TTYO    LDA 3,DMODE,2      ;GET MODE WORD
        MOVZR# 3,3,SZR     ;INPUT? OR OUTPUT?
        JMP  CHOT          ;OUTPUT, GO SERVICE

;INTERRUPT DUE TO ECHOING OF INPUT

        LDA 0,DCNT,2       ;PICKUP DATA COUNT VALUE
        MOV 0,0,SNR        ;CHECK IF FURTHER INPUT
        JMP  CHRO+3        ;NO,PERFORM I/O END
        LDA 2,DIDBO,2      ;GET INTERRUPT DATA BLOCK ADDR
        JMP  DISN          ;DISMISS INTERRUPT
```

```
;
;
;
;
;*****************************************************
;
;       INTERRUPT RE-ENTRANT ROUTINE TO HANDLE OUTPUT ON
;       TELETYPE AND SIMILAR DEVICES
;
;    C(AC2)=ADDR DEVICE UNIT CONTROL BLOCK
;    AC2,AC3 ALREADY SAVED
;
;*****************************************************
;
 CHOT   LDA 3,DCNT,2      ;GET DATA COUNT
        MOVL# 3,3,SZC     ;DATA COUNT > OR = 0?
        JMP  CHRO-1       ;NO => OUTPUT WAS TERMINATED
        MOV 3,3,SNR       ;GONE TO ZERO?
        JMP  CHRO         ;YES, I/O COMPLETE
        JSR OUT                 ;GO GET CHAR AND GENERATE IOT
        JMP  CHRO+1       ;RETURNS HERE IF I/O COMPLETE
        JSR DTEMP,2       ;EXECUTE INSTRUCTION
        LDA 2,DIDBO,2     ;GET INTERRUPT DATA BLOCK ADDR
        JMP  DISN         ;DISMISS INTERRUPT
;
;*****************************************************
;
;   ROUTINE TO GET CHARACTER AND GENERATE IO INSTRUCTION
;    C(AC2)=ADDRESS DEVICE UNIT CONTROL BLOCK
;
;*****************************************************
;
OUT     STA 3,DTEMP,2     ;SAVE RETURN ADDRESS
        JSR @DGSR,2       ;GET CHAR. WITH PROPER ROUTINE
        LDA 1,DCMDE,2     ;GET MODE WORD
        MOV 1,1,SZR       ;ASCII? OR IMAGE?
        JMP  +4                 ;IMAGE, IGNORE NULL AND PARITY
        MOV 0,0,SNR       ;ASCII, NULL CHARACTER?
        JMP OUT1          ;YES, I/O COMPLETE
        JSR @DPRTY,2      ;GENERATE PARITY
        LDA 3,DVCL,2      ;GET DEVICE CODE
        INC 3,3                 ;INCREMENT (CONVERTS TTI TO TTO)
        LDA 1, DOAS       ;GET INSTRUCTION
        ADD 3,1                 ;ADD DEVICE CODE
        ISZ DTEMP,2       ;INCREMENT FOR NORMAL RETURN
OUT1    LDA 3,DTEMP,2     ;RESTORE RETURN
        STA 1,DTEMP,2     ;SAVE IOT INSTRUCTION
        JMP 0,3                 ;EXIT
;
;OUTPUT OPERATION COMPLETE
;
        NEG 3,0,SKP       ;NEGATE TERMINATION CHAR.
 CHRO   ADC 0,0                 ;GENERATE -1
        LDA 3,DTCBA,2     ;GET ADDR CALLER'S QUEUE BLOCK
        STA 0,TAC3,3      ;SET RETURN AC3
        LDA 0,DIDBO,2     ;GET INTERRUPT DATA BLOCK ADDR
        JMP  IOEND        ;I/O OPERATION COMPLETE
```

```
;
;*******************************************************
;
;       INTERRUPT RE-ENTRANT ROUTINE TO HANDLE INPUT ON
;       TELETYPE AND SIMILAR DEVICES
;
;    C(AC2)=ADDR DEVICE UNIT CONTROL BLOCK
;    C(AC0)=INPUT CHARACTER
;    AC0,AC2,AC3 ALREADY SAVED
;
;*******************************************************
;
CHIN:   LDA 3,DCMDE,2     ;GET MODE WORD
        MOV 3,3,SZR       ;ASCII? OR IMAGE?
        JMP CHRI1         ;IMAGE. IGNORE SPECIAL CHECKS
        MOV 0,0,SNR       ;ASCII. CHECK FOR NULL
        JMP CHRI2         ;NULL => IGNORE
        JSR @DPRTY,2      ;CHECK PARITY
        COM# 0,0,SNR      ;ERROR?
        JMP CHRI3         ;YES. TAKE ERROR RETURN
        LDA 3,DTERM,2     ;GET  ADDR LIST OF TERMINATORS
        LDA 1,0,3         ;GET TERMINATOR CHARACTER CODE
        MOVZL# 1,1,SZC    ;END OF LIST?
        JMP CHRI1         ;YES. CONTINUE
        SUB 0,1,SNR       ;NO. CHECK FOR MATCH
        JMP .CHRI+1       ;TERMINATION CHAR => I/O COMPLETE
        INC 3,3                   ;GET NEXT ADDR
        JMP .-6                   ;TRY AGAIN
;
CHRI1:  JSR @DGSR,2       ;STORE CHAR. WITH PROPER ROUTINE
        LDA 3,DCNT,2      ;GET DATA COUNT
        MOV 3,3,SNR       ;GONE TO ZERO?
        JMP .CHRI         ;YES. I/O COMPLETE
;
CHRI2:  LDA 3,DVCDE,2     ;GET DEVICE CODE
        LDA 1,.NIOS       ;GET INSTRUCTION
        ADD 1,3                   ;ADD TO DEVICE CODE
        STA 3,DTEMP,2     ;SAVE INSTRUCTION
        JSR DTEMP,2       ;EXECUTE IT
DISN:   LDA 2,DIDBI,2     ;GET INTERRUPT DATA BLOCK ADDR
        JMP .DISN         ;DISMISS INTERRUPT
;
;TELETYPE INPUT INTERRUPT WHILE IN OUTPUT MODE
;
HLTOT:  LDA 1,DCNT,2      ;GET CURRENT WORD COUNT
        NEGZL# 1,1,SNC    ;IF <= 0 => IN TERMINATION STATE
        JMP DISN          ;SO DISMISS INPUT INTERRUPT
;
;OUTPUT TO BE TERMINATED
;
        NEG 0,1                   ;SET TERMINATION CHAR. NEGATIVE
        LDA 0,DERTN,2     ;GET ERROR RETURN ADDR.
        LDA 3,DTCBA,2     ;GET CALLER'S QUEUE BLOCK ADDR
        STA 1,DCNT,2      ;SAVE AC1 AS NEW WORD COUNT
        STA 0,TPC,3       ;SAVE AS RETURN PC ADDR.
        JMP DISN          ;DISMISS INPUT INTERRUPT
```

```
;ERROR RETURN

CHRI3   LDA 3,DTCBA,2     ;GET ADDR CALLER'S QUEUE BLOCK
        LDA 1,DERTN,2     ;GET ERROR RETURN ADDR
        STA 1,TPC,3       ;SAVE AS RETURN PC
        JMP  CHRI+2       ;I/O COMPLETE

 INPUT OPERATION COMPLETE

 CHRI   ADC 0,0,SKP       ;GENERATE -1
        STA 1,DCNT,2      ;SET WORD COUNT VALUE 0
        LDA 3,DTCBA,2     ;GET ADDR CALLER'S QUEUE BLOCK
        STA 0,TAC3,3      ;SET RETURN AC3
        LDA 0,DMODE,2     ;GET MODE WORD
        MOVR 0,0,SNC      ;SKIP IF NOT ECHOING
        JMP DISN          ;DISMISS INPUT INTERRUPT
        LDA 0,DIDBI,2     ;GET INTERRUPT DATA BLOCK ADDR
        JMP  IOEND        ;I/O OPERATION COMPLETE


 DOAS   DOAS 0,0
 NIOS   NIOS 0

;*****************************************************
;
;      ROUTINE TO DISMISS INTERRUPT
;    C(AC2)=ADDR DATA BLOCK IN INTERRUPT SERVICE ROUTINE
;
;*****************************************************

 DISN   LDA 0,IAC0,2      ;RESTORE AC0
        LDA 1,IAC1,2      ;RESTORE AC1
       LDA 3,IOMSK,2     ;RECALL ORIGINAL PRIORITY WORD
        DOBC 3 CPU        ;RESTORE IT HARDWARE-INTDS
        STA 3,@PPMSK      ;RESTORE IT -- SOFTWARE
        LDA 3,IPC,2       ;GET RETURN PC
        MOV 3 3 SNR       ;WAS A NULL EXECUTING ?
         QUIT             ;YES--RESCHEDULE
        STA 3,0                  ;SAVE RETURN PC
        LDA 3 ICRY 2      ;GET CARRY WORD
        MOVL 3 3          ;RESTORE CARRY
        LDA 3,IAC3,2      ;RESTORE AC3
        LDA 2,IAC2,2      ;RESTORE AC2
        INTEN             ;RE-ENABLE INTERRUPT
        JMP @0            ;RETURN TO INTERRUPTED PROGRAM

PPMSK    PMSK             ;POINTER TO CURRENT PRIORITY
```

```
;*****************************************************
;
;       ROUTINE TO HANDLE END OF I/O OPERATION
;
;    C(AC0)=ADDRESS INTERRUPT DATA BLOCK
;         (0 IF NON-INTERRUPT)
;    C(AC2)=ADDRESS DEVICE UNIT CONTROL BLOCK
;
;*****************************************************
IOEND:          LDA 1,DQBSY,2    ;FETCH QUEUE AVAILABILITY SWITCH
        MOV 1,1,SZR       ;QUEUE AVAILABLE?
        JMP  END3         ;NO--CREATE NEW JOB (INTERRUPT MODE)
        LDA 1,@DTCBA,2    ;YES, GET ADDR NEXT I/O REQUEST
        STA 0,DTEMP,2     ;SAVE INTERRUPT DATA ADDRESS
        MOV 2,0                   ;SAVE AC2
        LDA 2 DTCBA 2     ;FETCH ADDR OF JOB TO BE ACTIVATED
        JSR @RPSH         ;ACTIVATE IT
        MOV 1,1,SZR       ;ANOTHER REQUEST PENDING?
        JMP  END4         ;YES, CREATE JOB TO DO NEXT I/O
        MOV 0,2                   ;NO,RESTORE DUCB ADDR TO AC2
        STA 1,DFLNK,2     ;NO, MAKE DEVICE AVAILABLE
END0:   LDA 2,DTEMP,2     ;GET INTERRUPT DATA BLOCK ADDR
        MOV 2,2,SNR       ;INTERRUPT EXIT?
         QUIT             ;NO, TERMINATE JOB
END1:   LDA 1,.SYS.       ;YES, PICKUP MODE SWITCH
        MOV 1,1,SZR       ;SYSTEM MODE?
        JMP .DISN         ;YES-DISMISS INTERRUPT AND CONTINUE
        LDA 0 IOMSK 2     ;PICKUP OLD PRIORITY MASK
        LDA 1 IPC 2       ;(USER MODE)--PICKUP RETURN PC
        MOV 1 1 SNR       ;WAS PROGRAM COUNTER = 0 ?
        JMP .END2         ;YES => NULL JOB WAS OPERATIONAL
        MOV 0 0 SZR       ;NO--CHECK OLD HARDWARE MASK
        JMP .DISN         ;NON-ZERO => INTERRUPTED ANOTHER
                          ;INTERRUPT SERVICING PROGRAM
        MOV 2,0                   ;ZERO--SAVE INTERRUPT DATA ADDR
        JSR @PPOP         ;GET QUEUE BLOCK
        MOV 0,3                   ;RESTORE INTERRUPT DATA ADDR
        LDA 1,C377        ;GET PRIORITY MASK
        LDA 0,@PCPTY      ;GET CURRENT PRIORITY
        AND 1,0                   ;RESTRICT PRIORITY TO 8 BITS
        LDA 1,ICRY,3      ;GET CARRY IN AC1 BIT 0
        ADD 0,1                   ;HAVE QUEUE ENTRY
        STA 1,TPTY,2      ;ENTER IT
        LDA 0,IAC0,3      ;GET AC0
        STA 0,TAC0,2      ;ENTER IT
        LDA 0,IAC1,3      ;GET AC1
        STA 0,TAC1,2      ;ENTER IT
        LDA 0,IAC2,3      ;GET AC2
        STA 0,TAC2,2      ;ENTER IT
        LDA 0,IAC3,3      ;GET AC3
        STA 0,TAC3,2      ;ENTER IT
        LDA 0,IPC,3       ;GET RETURN PC
        STA 0,TPC,2       ;ENTER IT
        LDA 1,IOMSK,3     ;GET OLD PRIORITY MASK WORD
        STA 1,@PPMSK      ;RESTORE IT --- SOFTWARE
        MSKO 1            ;RESTORE IT --- HARDWARE
        JSR @RPSH         ;ACTIVATE JOB
         QUIT             ;EXIT TO SCHEDULER
```

```
;NULL JOB WAS INTERRUPTED
;ENTERED WITH:
;        AC0=OLD HARDWARE PRIORITY MASK

 END2    STA 0,@PPMSK      ;RESTORE PRIORITY...SOFTWARE
         MSKO 0            ;RESTORE PRIORITY   HARDWARE
          QUIT             ;EXIT TO THE SCHEDULER

;CREATE JOB OF PRIORITY 0 TO PERFORM END
;OF OPERATION AT NON-INTERRUPT LEVEL BECAUSE
;QUEUE FOR DEVICE WAS NOT AVAILABLE AT THE
;TIME OF THE INTERRUPT

 END3:   MOV 2,1                       ;SAVE AC2
         JSR @PPOP         ;GET QUEUE BLOCK
         STA 1,TAC2,2      ;ENTER AC2
         SUB 1,1
         STA 1,TPTY,2      ;RETURN PRIORITY=0
         STA 1,TAC0,2      ;RETURN AC0=0
         LDA 1,NEWPC       ;PICKUP RETURN ADDR
         STA 1,TPC,2       ;ENTER AS RETURN PC
         JSR @RPSH         ;ACTIVATE JOB
         MOV 0,2                       ;RESTORE INTERRUPT DATA ADDR
         JMP .DISN         ;DISMISS INTERRUPT

NEWPC:    IOEND            ;RETURN ADDR

;CREATE JOB OF PRIORITY ZERO TO START
;NEXT I/O OPERATION ON DEVICE

 END4:   JSR @PPOP         ;GET QUEUE BLOCK
         STA 0,TAC3,2      ;SET AC3
         STA 1,TAC2,2      ;SET AC2
         MOV 1,3                       ;RESTORE QUEUE BLK ADDR TO AC3
         LDA 3,TAC3,3      ;FETCH CALLER'S AC3(ADDR .IOX+1)
         LDA 1,ICNTL,3     ;PICKUP DEVICE CONTROL WORD
         STA 1,TAC0,2      ;ENTER AS RETURN AC0
         SUB 1,1                       ;CLEAR AC1
         STA 1,TAC1,2      ;SET AS RETURN AC1
         STA 1,TPTY,2      ;SET AS RETURN PRIORITY
         MOV 0,3                       ;GET DUCB ADDR IN AC3
         LDA 1,DNIOR,3     ;GET ADDR NEXT I/O ROUTINE
         STA 1,TPC,2       ;SET AS RETURN PC
         JSR @RPSH         ;ACTIVATE JOB
         MOV 0 2                       ;GET DUCB ADDR. IN AC3
         JMP  ENDO         ;DETERMINE TYPE OF EXIT REQUIRED
                           ;   (INTERRUPT OR USER)
PPOP:    QSPOP   ;SUBROUTINE POINTER
RPSH:    JSPSH   ;SUBROUTINE POINTER
PCPTY:    CPTY   ;POINTER TO CURRENT PRIORITY STORAGE
```

```
;
;*****************************************************
;
;       THE FOLLOWING ROUTINES ARE USED BY THE
;       HANDLERS TO GENERATE/TEST ODD OR EVEN PARITY
;
;CALLING SEQUENCES:
;       LDA 0,CHAR7      ;GET 7 BIT CHAR IN AC0
;       JSR (GODD,GEVEN);GENERATE ODD OR EVEN PARITY
;       <RETURN>         ;CHAR IN AC0 WITH PARITY BIT SET
;
;       LDA 0,CHAR8      ;GET 8 BIT CHAR IN AC0
;       JSR (TODD,TEVEN);TEST ODD OR EVEN PARITY
;       <RETURN>         ;PARITY OK = 7 BIT CHAR IN AC0
;                        ;PARITY FAILURE => C(AC0)=-1
;
;ON ENTRY
;C(AC2)=ADDR DEVICE UNIT CONTRL BLOCK
;
;ALL ROUTINES ARE RE-ENTRANT
;
;*****************************************************
;
GODD:   ADDOR 2,2,SKP    ;SET AC2 BIT 0 FOR GENERATION
GEVEN:  ADDOR 2,2,SKP    ; "   "   "   "   "       "
TODD:   ADDOR 3,3        ;SET AC3 BIT 0 FOR ODD PARITY
TEVEN:  MOVZS 0,1        ;CEAR CARRY AND SHIFT LEFT 8
        ADD 1,1,SZR      ;SHIFT LEFT AND COMPUTE PARITY
        JMP .-1                  ;REPEAT U AL BITS TALLIED
        LDA 1,BIT0       ;FETCH 100000 IN AC1
        ADD 1,3                  ;COMPLEMENT CARRY IF ODD DESIRED
        MOVL# 2,2,SZC    ;TESTING OR GENERATING?
        JMP .+5                  ;GENERATING--GO D IT
        LDA 1,C177       ;GET 7-BIT MASK
        AND 1,0,SZC      ;CONVERT TO 7-BIT; CHECK PARITY
        ADC 0,0                  ;PARITY ERROR
        JMP 0,3                  ;RETURN
        ADD 1,2                  ;CLEAR AC2 BIT 0
        MOVS 1,1,SNC     ;CREATE 200, SKIP IF NOT REQ'D
        ADD 1,0                  ;ADD IN PARITY BIT
        JMP 0,3                  ;RETURN
BIT0    80               ;100000
```

```
;*******************************************************
;
;       THE FOL[illegible] ROUTINES GET/STORE AN 8 BIT
;       CHARA[illegible] [illegible]E 1[illegible] WORD
;
;
;    [illegible]ADDR DEVICE UNIT CONTROL BLOCK
;
;*******************************************************
GTCHR:  LDA 0,@DDADR,2     ;PICKUP CHARACTER
        LDA 1,C377         ;GET AN 8 BIT MASK
        AND 1,0,SKP        ;MASK

STCHR:  STA 0,@DDADR,2     ;STORE CHARACTER
        ISZ DDADR,2        ;INCREMENT POINTER
        DSZ DCNT,2         ;DECREMENT DATA COUNT
        JMP 0,2                    ;EXIT -- NORMAL
        JMP 0,3                    ;EXIT -- COUNT GONE TO ZERO

;*******************************************************
;
;       THE FOLLOWING ROUTINES PACK AND UNPACK 2 8-BIT
;       CHARACTERS PER 16 BIT WORD
;
;    ON ENTRY
;    C(AC2)=ADDR DEVICE UITCNRO BLOC
;
;*******************************************************
GCHRP:  MOV 3,1                    ;SAVE RETURN ADDR
        LDA 3,DDADR,2      ;GET DATA POINTER
        MOVZR 3,3          ;CONVERT TO ADDR
        LDA 0,0,3          ;PICKUP WORD
        MOV 0,0,SZC        ; RH? OR LH?
        MOVS 0,0           ;LH => SWAP
        LDA 3,C377         ;GET 8 BIT MASK
        AND 3,0                    ;HAVE CHARACTER
        MOV 1,3                    ;RESTORE RETURN
        JMP STCHR+1        ;EXIT

SCHRP:  STA 3,DTEMP,2      ;SAVE RETURN ADDR
        LDA 3,DDADR,2      ;GET POINTER
        MOVZR 3,3          ;CONVERT TO ADDRESS
        LDA 3,0,3          ;GET WORD FROM STORAGE
        MOV 0,0,SZC        ;TEST WHICH HALF
        MOVS 3,3           ;LEFT HALF
        LDA 1,M377         ;GET 8 BIT MASK TO LEFT HALF
        AND 3,1,SZC        ;MASK AND TEST
        ADDS 0,1,SKP       ;INTO LEFT HALF
        ADD 0,1                    ;INTO RIGHT HALF
        LDA 3,DDADR,2      ;GET POINTER AGAIN
        MOVZR 3,3          ;CONVERT TO ADDRESS
        STA 1,0,3          ;STORE UPDATED WORD
        LDA 3,DTEMP,2      ;RESTORE RETURN
        JMP STCHR+1        ;EXIT
```

```
;
;MASK AC0 TO 7 BITS AND EXIT
;

BIT7:   LDA 1,C177
        AND 1,0
        JMP 0,3

;
;SET CHANNEL 8 TO ONE
;

CHN8    LDA 1,C200      ;GET 200
        AND# 1,0,SNR    ;ALREADY SET?
        ADD 1,0                 ;NO, ADD IT IN

        JMP 0,3                 ;YES, EXIT

C177:   177             ;7 BIT MASK
M377:   177400          ;LEFT 8 BIT MASK
C200:   200             ;BIT 8 ON ONLY
C377:   377             ;RIGHT 8 BIT MASK

****
;
;       ROUTINE TO SETUP DEVICE UNIT CONTROL BLOCK
;    C(AC3)=ADDR DEVICE CONTROL BLOCK
;    C(AC0)=CALLER'S DEVICE CONTROL WORD
;
;*****************************************************

DUCB:   SUB 1,1                 ;CLEAR AC1
        STA 1,DCMDE,3   ;ZERO ASCII/IMAGE MODE WORD
        MOVZL 0,0,SZC   ;INPUT? OR OUTPUT?
        INC 1,1,SKP     ;OUTPUT, GENERATE +2, (NO ECHO)
        MOVZL# 0,0,SZC  ;INPUT, CHECK FOR ECHO
        INC 1,1                 ;MAKE 1 FOR NO ECHO, 2 FOR OUTPUT
        STA 1,DMODE,3   ;ENTER MODE IN CONTROL BLOCK
        SUB 2,2                 ;CLEAR AC2
        ADDZL 0,0,SZC   ;CHECK WORD/CHAR FORMAT BIT
        INCZL 2,2       ;WORD FORMAT, GENERATE +2
        MOVZR# 1,1,SZR  ;INPUT? OR OUTPUT?
        INC 2,2                 ;OUTPUT
        LDA 1, BTAB     ;GET BASE ADDR BYTE ROUTINE TABL
        ADD 1,2                 ;HAVE POINTER TO CORRECT ROUTINE
        LDA 1,0,2       ;GET ADDR ROUTINE
        STA 1,DGSR,3    ;ENTER IN UNIT CONTROL BLOCK
        MOVZL 0,0,SZC   ;ASCII? OR IMAGE?
        ISZ DCMDE,3     ;IMAGE
        SUB 2,2                 ;CLEAR AC2 AGAIN
        MOVZL 0,0,SZC   ;CHECK CONTROL WORD BIT 4
```

```
        INCZL 2,2          ;ONE, INCREMENT & MULTIPLY BY 2
        MOVZL 0,0,SZC      ;CHECK CONTROL WORD BIT 5
        INCZL 2,2,SKP      ;ONE, INCREMENT & MULTIPLY BY 2
        MOVZL 2,2          ;ZERO, JUST MULTIPLY BY 2
        LDA 1,DMODE,3      ;GET I/O MODE FROM DUCB
        MOVZR# 1,1,SZR     ;INPUT? OR OUTPUT?
        INC 2,2                    ;OUTPUT
        LDA 0, PTAB        ;GET BASE ADDR PARITY ROUTINES
        ADD 0,2                    ;HAVE POINTER TO CORRECT ROUTINE
        LDA 0,0,2          ;GET ADDRESS PARITY ROUTINE
        STA 0,DPRTY,3      ;ENTER IN UNIT CONTROL BLOCK
DCB1:   LDA 2,DTCBA,3      ;GET CALLER'S QUEUE BLOCK ADDR
        LDA 2,TAC3,2       ;RESTORE AC3 (IE ADDR . IOX+1)
        LDA 0,IDPTR,2      ;GET DATA POINTER
        STA 0,DDADR,3      ;ENTER IN UNIT CONTROL BLOCK
        LDA 0,IDCNT,2      ;GET DATA COUNT
        MOV 0 0 SNR        ;IS IT GREATER THAN ZERO?
        JMP SET5           ;NO, => NO I/O TO DO
        STA 0,DCNT,3       ;YES, ENTER IN UNIT CONTROL BLOCK
        LDA 0,DVCDE,3      ;GET DEVICE CODE
        MOVZR# 1,1,SZR     ;INPUT OR OUTPUT MODE?
        JMP SET3           ;OUTPUT
        LDA 2,.NIO         ;GET I/O INSTRUCTION
        ADD 0,2                    ;ADD IN DEVICE CODE
        STA 2,DTEMP,3      ;SAVE IT
        JSR DTEMP,3        ;EXECUTE IT
        .QUIT              ;EXIT TO SCHEDULER

;OUTPUT I/O MODE

SET3:   MOV 3,2                    ;GET DUCB ADDR INTO AC2
        JSR @POUT          ;GET 1ST I/O AND GENERATE INSTR.
        JMP SET4           ;RETURNS HERE IF I/O COMPLETE
        JSR DTEMP,2        ;EXECUTE THE INSTRUCTION
        .QUIT              ;EXIT TO SCHEDULER

;I/O OPERATION WAS COMPLETE

;COMPLETED BECAUSE BUFFER CONTAINED NULL
;AS FIRST CHARACTER WHILE IN ASCII MODE

SET4:   MOV 2 3                    ;SET AC3 = DUCB ADDRESS
        SUB 0 0                    ;MARK NON-INTERRUPT IOEND CALL
        SUB 1 1 SKP        ;SET RETURN AC3 = 0

;COMPLETED BECAUSE INITIAL WORD COUNT WAS ZERO

SET5    ADC 1 1                    ;SET RETURN AC3=-1
        LDA 2 DTCBA 3      ;SET AC2 TO TCB ADDR.
        STA 1 TAC3 2       ;SET RETURN AC3
        MOV 3,2                    ;GET DUCB ADDR INTO AC2
        JMP @QEND          ;I/O IS COMPLETE

POUT    OUT                ;SUBROUTINE POINTER
QEND     IOEND             ;POINTER TO IOEND ROUTINE

 NIO    NIOS 0             ;INSTRUCTION CONSTANTS
```

```
;***************************************************************
;        GENERAL PURPOSE TELETYPE HANDLER

;THIS HANDLER ALLOWS FOR THE SIMULTANEOUS OPERATION OF
;A VARIABLE NUMBER OF ASR-33 TELETYPES
;
;EACH TELETYPE IS TREATED AS A SINGLE DEVICE OPERABLE
;IN ANY ONE OF THREE MODES.
;        1  INPUT
;        2  INPUT (SUPPRESS ECHO)
;        3  OUTPUT
;
;
;IN MODES 1 AND 2, INTERRUPTS FROM TTO ARE IGNORED.
;
;IN MODE 3, AN INTERRUPT FROM TTI CAUSES TERMINATION OF
;OUTPUT   IF THE BREAK CHARACTER CAUSED THE INTERRUPT
;.ALL TASKS WITH TELETYPE I/O ACTIVE AREA MADE DORMANT
;AND IF THE TTY UNIT WAS ENABLED TO THE BREAK
;FACILITY, THE BREAK JOB IS ACTIVATED.
;OTHERWISE, THE ERROR RETURN IS TAKEN WITH THE 8 BIT
;CODE OF THE CHARACTER CAUSING TERMINATION RETURNED
;IN AC3.
;
;******************************************************
;
;       ENTRY POINT FROM .IOX INSTRUCTION CALL
;    AC2 CONTAINS ADDR OF CREATED QUEUE BLOCK
;    AC0 CONTAINS TTY CONTROL WORD
;
;******************************************************

TTYIO:  LDA 1,C377      ;GET AN 8 BIT MASK
        AND 0,1                 ;HAVE UNIT# IN AC1
        LDA 3,TTNO      ;GET NUMBER OF UNITS IN SYSTEM
        SUBZ# 3,1,SZC   ;REQUESTED UNIT EXISTS?
        JMP @UNERR      ;NO, UNIT ERROR
        LDA 3,TTBLK     ;YES, GET BASE ADDR DUCB TABLE
        ADD 1,3                 ;HAVE ADDR POINTER TO DUCB
        LDA 3,0,3       ;GET ADDR  UNIT CONTROL BLOCK
        ISZ DQBSY,3     ;INDICATE QUEUE BUSY
        LDA 1,DTCBA,3   ;GET FIRST ENTRY
        MOV 1,1,SZR     ;UNIT AVAILABLE?
        JMP IOSTK       ;NO, GO STACK I/O REQUEST

 TTIO:  STA 2,DTCBA,3   ;ENTER QUEUE ADDR IN DUCB
        STA 1,DQBSY,3   ;INDICATE I/O QUEUE AVAILABLE
        LDA 2,TAC3,2    ;RESTORE ADDR .IOX+1
        LDA 1,IERTN,2   ;GET ERROR RETURN ADDRESS
        STA 1,DERTN,3   ;ENTER IT IN DUCB
        JMP .DUCB       ;FINISH DUCB SETUP AND BEGIN I/O
UNERR:  .UNER           ;ADDRESS FOR UNIT ERROR
        .DVER           ;ADDRESS FOR DEVICE UNIT ERROR

;CONSTANT CONTAINING THE NUMBER OF TELETYPES ON SYSTEM
TTNO:   TTYS
```

```
;
;****************************************************
;
;       ENTER HERE IF I/O ALREADY IN PROGRESS
;          (QUEUE BUSY SWITCH SET)
;    C(AC3)=ADDR DEVICE UNIT CONTROL BLOCK
;    C(AC2)=ADDR CALLER'S QUEUE BLOCK
;    C(AC1)=ADDR QUEUE BLOCK CURRENTLY BEING PROCESSED
;
;****************************************************
;
IOSTK   MOV 2,0                  ;SAVE CALLER'S QUEUE ADDR IN AC0
        MOV 1,2                  ;RESTORE CURR. QUEUE ADDR TO AC2
        LDA 1,DFLNK,2    ;GET FWD LINK
        MOV 1,1,SZR      ;END OF QUEUE?
        JMP .-3                  ;NO, CONTINUE
        STA 0,DFLNK,2    ;YES, ADD BLK TO END OF QUEUE
        STA 1,DQBSY,3    ;INDICATE QUEUE AVAILABLE
        JMP @0SCHD       ;EXIT TO SCHEDULER
;
;
;
;****************************************************
;
;       THE FOLLOWING IS A TABLE OF GET/STORE CHARACTER
;       SUBROUTINE ADDRESSES
;
;****************************************************
BTAB:   .+1
        SCHRP   ;STORE CHARACTER (BYTE MODE)
        GCHRP   ;GET CHARACTER (BYTE MODE)
        STCHR   ;STORE CHARACTER (WORD MODE)
        GTCHR   ;GET CHARACTER (WORD MODE)
;
;****************************************************
;
;       THE FOLLOWING IS A TABLE OF PARITY
;       CHECKING/GENERATION SUBROUTINE ADDRESSES
;
;****************************************************
;
PTAB:   .+1
        BIT7    ;MASK TO 7 BITS (INPUT)
        BIT7    ;MASK TO 7 BITS (OUTPUT)
        TEVEN   ;CHECK EVEN PARITY
        GEVEN   ;GENERATE EVEN PARITY
        TODD    ;TEST ODD PARITY
        GODD    ;GENERATE ODD PARITY
        BIT7    ;MASK TO 7 BITS (INPUT)
        CHN8    ;SET CHANNEL 8 TO ONE (OUTPUT)
```

```
;****************************************************************
;
;       TELETYPE UNIT 0 DEVICE CONTROL BLOCK
;
;****************************************************************

TTY0:   0                       ;ADDR QUEUE BLOCK, 0 IF INACTIVE
        0                       ;VARIABLE LOCATION
        JMP 0,3                 ;SUBROUTINE RETURN INSTRUCTION
        0                       ;ADDR GET/STORE CHAR ROUTINE
        10                      ;DEVICE CODE TTI
        0                       ;DATA POINTER
        0                       ;DATA COUNT
        0                       ;0-QUEUE AVAILABLE 1-QUEUE BUSY
        0                       ;0-ASCII MODE 1-IMAGE MODE
        0                       ;ADDR PARITY ROUTINE
        .TTIO                   ;ADDR "NEXT I/O" ROUTINE
        TTO0+3                  ;ADDR INTERRUPT DATA BLOCK (OUT)
        .TERM                   ;ADDR LIST OF INPUT TERMINATORS
        TTI0+3                  ;ADDR INTERRUPT DATA BLOCK (INP)
        0                       ;ERROR RETURN ADDR
        0                       ;TELETYPE OPERATING MODE (0-2)
        0                       ;-1=>BREAK DISABLED, ELSE UNIT#
;
;TABLE OF DEVICE UNIT CONTROL BLOCKS
;FOR TELETYPES ATTACHED TO THE SYSTEM
;
TTBLK:   +1
         TTY0   ;DUCB OF TELETYPE UNIT 0
         IFE TTYS-2
         EXTN  TTY1,.TTI1,.TTO1
         TTY1   ;DUCB OF TELETYPE UNIT 1
         ENDC
;
QUE:    0         ;ADDRESS OF FIRST ACTIVE TCB
;
;INPUT TERMINATOR LIST (NO PARITY)
;
.TERM:  177       ;DEL ,RUB OUT
        15        ;CR-CARRIAGE RETURN; CNTRL M
        12        ;LF-LINE FEED
        33        ;ESC-ESCAPE
        3         ;ETX-END OF TEXT;EOM-END OF MESSAGE;CNTRL C
        -1
```

```
;*****************************************************************
;
;        ADDRESSES OF THE DEVICE HANDLERS
;
;        ORIENTATED IN LOGICAL DEVICE CODE ORDER
;        (ORDER CAN BE DETERMINED BY THE USER)
;*****************************************************************

HANTB    TTIO              ; DEVICE 0

          IFN HSR
          EXTN .PTR
          PTR              ; DEVICE 1
          ENDC
          IFE HSR
         IOBAD
          ENDC

          IFN HSP
          EXTN .PTP
          PTP              ; DEVICE 2
          ENDC
          IFE HSP
         IOBAD
          ENDC

          IFN PRINT
          EXTN .LPT
          LPT              ; DEVICE 3
          ENDC
          IFE PRINT
         IOBAD
          ENDC

          IFN PLOT
          EXTN .PLT
          PLT              ; DEVICE 4
          ENDC
          IFE PLOT
         IOBAD
          ENDC


          IFN CARD
          EXTN .CDR
          CDR              ; DEVICE 5
          ENDC
          IFE CARD
         IOBAD
          ENDC

          IFN DISK
          EXTN .DSK
          DSK     F0       ; DEVICE 6
          ENDC
```

```
 IFE DISK
IOBAD
 ENDC

 IFN A2D
 EXTN  ADCV
 ADCV              ; DEVICE 7
 ENDC
 IFE A2D
IOBAD
 ENDC

 IFN TAPE
 EXTN  MTA
 MTA               ; DEVICE 10(8)
 ENDC
 IFE TAPE
IOBAD
 ENDC

 IFN DCOM
 EXTN  DCOM
 DCOM              ; DEVICE 11(8)
 ENDC
 IFE DCOM
IOBAD
 ENDC

 IFN DPACK
 EXTN  DKP
 DKP               ;DEVICE 12 (8)
 ENDC
 IFE DPACK
IOBAD
 ENDC

 IFN QMUX
 EXTN  QMUX
 QMUX              ;DEVICE 13(8)
 ENDC
 IFE QMUX
IOBAD
 ENDC

 IFN IBM
 EXTN  IBM
 IBM               ; DEVICE 14(8)
 ENDC
 IFE IBM
IOBAD
 ENDC

 IFN JRB           ; DEVICE 15(8)
 EXTN  JRBI
 JRBI
 ENDC
 IFE JRB
IOBAD
 ENDC
```

AD-A096 421 ARMY AVIATION RESEARCH AND DEVELOPMENT COMMAND ST LO--ETC F/G 17/7
THE DEVELOPMENT AND TESTING OF THE NAVSTAR GLOBAL POSITIONING S--ETC(U)
FEB 81 J GRAY
UNCLASSIFIED USAAVRADCOM-TR-80-E-3 NL

2 OF 2

AD A
096421

END
DATE
FILMED
4-81
DTIC

```
;
;
 IFN BUTN          ;DEVICE 16(8)
 EXTN  BUTI        ;
 BUTI


 ENDC              ;
 IFE BUTN          ;
IOBAD              ;
 ENDC              ;
 IFN    SERIO             ;DEVICE 17(8)
 EXTN    SIOO
 SIOO
 ENDC
 IFE    SERIO
IOBAD
 ENDC
 IFN    RCU               ;DEVICE 20(8)
 EXTN   IRCUO
IRCUO
 ENDC
 IFE    RCU
IOBAD
 ENDC
 IFN    KW7S          ;DEVICE 21(8)
 EXTN    KW7
 KW7
 ENDC
 IFE    KW7S
IOBAD
 ENDC
 IFN HDMA1
 EXTN  DMA1
 DMA1
 ENDC
 IFE HDMA1
IOBAD
 ENDC
 IFN HDMA2
 EXTN  DMA2
 DMA2
 ENDC
 IFE HDMA2
IOBAD
 ENDC
;
HANTE:    BLK 0         ;END OF HANDLER TABLE
                        ;MUST BE IMMEDIATELY AT THE END
                        ;OF THE DEVICE HANDLER DEFINITIONS
;
IOBAD=  UNERR+2                 ;IOBAD-1 MUST BE THE ENTRY TO THE
                        ;DEVICE UNIT NUMBER ERROR ROUTINE
```

```
;*****************************************************
;
;DEVICE INTERRUPT SERVICING ROUTINE ENTRY POINTS
;
;IT IS A DISPATCH TABLE FOR THE INTERRUPT PROCESSOR
;
;*****************************************************

DISP     DUNG
         DUNG

          IFN JRB           ;JRB UNIT 0 INPUT(CODE 2)
          EXTN  JRBO        ;JRB UNIT 0 OUTPUT(CODE 3)
          JRBI
          JRBO
          ENDC
         IFE JRB
         NODEV
         NODEV
         ENDC

         .IFN      FGATE
         .EXTN     GATE
         GATE
          ENDC
         .IFE      FGATE
         NODEV
         .ENDC
         NODEV

NODEV
.IFN HDMA1
 DMA1
.ENDC
 IFE HDMA1
NODEV
 ENDC

         TTIO               ;TTY UNIT 0 INPUT(CODE 10)
         TTOO               ;TTY UNIT 0 OUTPUT(CODE 11)

          IFN HSR
          PTR               ;PTR SERVICE(CODE 12)
         .ENDC
          IFE HSR
         NODEV
         .ENDC

          IFN HSP
         .PTP               ;PTP SERVICE(CODE 13)
          ENDC
         .IFE HSP
         NODEV
          ENDC
```

```
CKSER                  ;CLOCK SERVICE(CODE 14)

 IFN PLOT
 PLT                   ;PLOTTER SERVICE(CODE 15)
 ENDC
 IFE PLOT
NODEV
 ENDC

 IFN CARD
 CDR                   ;CARD READER(CODE 16)
 ENDC

 IFE CARD
NODEV
 ENDC


  IFN PRINT
  LPT                  ;LINE PRINTER(CODE 17)
  ENDC
  IFE PRINT
 NODEV
  ENDC

  IFN DISK
  DSK                  ;DISK SERVICE(CODE 20)
  ENDC
  IFE DISK
 NODEV
  ENDC

  IFN A2D
  ADCV                 ;A/D SERVICE(CODE 21)
  ENDC
  IFE A2D
 NODEV
  ENDC

  IFN TAPE
  MTA                  ;MAG TAPE(CODE 22)
  ENDC
  IFE TAPE
 NODEV
  ENDC

 NODEV                 ;NO INTERRUPT WITH D2A(CODE 23)

  IFN DCOM
  DCOM                 ;DATA COMMUNICATION (CODE 24)
  ENDC
  IFE DCOM
```

```
        NODEV
         ENDC


        NODEV           ;CODE 25
        NODEV           ;CODE 26
        NODEV           ;CODE 27

         IFN QMUX
         QMUX           ;TYPE 4060 MULTIPLEXOR CODE 30
        .ENDC
         IFE QMUX
        NODEV
         ENDC

        .IFN IBM
         IBM            ;TYPE 4025 NOVA-SYSTEM 360 INTERFACE
        .ENDC
        .IFE IBM
       NODEV
        .ENDC
        NODEV           ;NO INTERRUPT FOR IBM2 PART OF 4025

        .IFN DPACK
        .DKP            ;MOVING HEAD DISK HANDLER CODE 33
        .ENDC
        .IFE DPACK
        NODEV
        .ENDC

        NODEV           ;CODE 34 - 37
        NODEV
        NODEV
        NODEV

        NODEV           ;CODE 40
        .IFN BUTN       ;PUSH BUTTON INTERFACE (CODE 41)
        .BUTI
         ENDC
        IFE BUTN
        NODEV
        .ENDC

        NODEV           ;CODE 42
        .IFN BUTN-1     ;SECOND PUSH BUTTON INTERFACE (CODE 43)
        .EXTN .BUII
        .BUII
         ENDC
        .IFE BUTN-1
        NODEV
        .ENDC
        NODEV           ;CODES 44 - 47
        NODEV
IFN HDMA2
DMA2
ENDC
IFE HDMA2
```

```
NODEV
 ENDC
NODEV                                                                    .   20400
 IFE    TTYS-2
 TTI1                    ;SECOND TELETYPE INPUT (CODE 50)
 TTO1                    ;SECOND TELETYPE OUTPUT (CODE 51)
 ENDC
 IFN    TTYS-2
NODEV
NODEV
 ENDC
NODEV
NODEV
 IFN    SERIO            ;OF CONTROL UNIT (CODE 54)
 SIO0




 ENDC
 IFE    SERIO
NODEV
 ENDC
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
NODEV
 IFN    KW7S         ;KW7 DEVICE CODE 70(8)
 KW7
 ENDC
 IFE    KW7S
NODEV
 ENDC
 IFN    RCU-1            ;RECEIVER CONTROL UNIT #1 (CODE 73
 EXTN   IRCU1
IRCU1
 ENDC
 IFE    RCU-1
NODEV
 ENDC
 IFN    RCU              ;RECEIVER CONTROL UNIT #0 (CODE 72
IRCU0
 ENDC
 IFE    RCU
NODEV
 ENDC
NODEV
NODEV
NODEV
NODEV                                                                    .   18930
DUNG                                                                         20510
```

```
;DEVICE CODE 77 (CPU) IS HANDLED ELSEWHERE
;IF THE POWER FAILURE MONITOR OPTION XX06
;WAS SPECIFIED IN THE HARDWARE CONFIGURATION
;DATA ON THE PARAMETER TAPE


;*******************************************
;
;       ENTRY POINT FOR  ENQU SYSTEM CALL
;       FORMAT
;        ENQU
;       ADDRESS> ADDRESS OF RCB
;       <RETURN>
;
;*******************************************

ENQ     ISZ .SYS.        ;INDICATE SYSTEM MODE
        STA 2,.AC2       ;STORE AC2 IN .AC2
        INC 3,3          ;ADJUST LINK REGISTER FOR RETURN
        STA 3,.RTN.      ;AND SAVE IT IN .RTN.
        JSR @ENDEQ       ;GO CREATE A QUEUE BLOCK
        STA 2,QUET1      ;TEMP SAVE TCB ADDR
        SUB 3,3          ;CLEAR FWD. LINK OF QUEUE ENTRY
        STA 3,TLINK,2
        LDA 3,.RTN.      ;OBTAIN ADDR. TO LOCK WORD
        LDA 3,-1,3       ;PICK UP ADDR OF RCB
        LDA 2,0,3        ;PICK UP THE RCB
        MOVZL 2,2,SZC    ;IS RESOURCE CLAIMED
        JMP ENQ1         ;YES, QUEUE THE REQUEST
        SUBZR 2,2        ;NO, SET CLAIM BIT
        STA 2,0,3        ;STORE IT BACK IN RCB
        LDA 2,QUET1      ;PICK UP ADDRESS OF TCB
        JMP @ENDEQ+1     ;ENTER SCHEDULER
;
;       ENTER TASK INTO RESOURCE DEP SUSP QUEUE
;       QUEUE IS MAINT  BY ORDER OF PRI OF SUSP TASK
;
ENQ1    MOV 2,2,SZR      ;ARE THERE TASKS WAITING
        JMP ENQ2         ;YES, GO ENTER TASK IN SUSP QUEUE
        LDA 2,QUET1      ;NO, GO PICK UP ADDR OF NEW TCB
        ADDOR 2,2        ;REINSERT CURRENT CLAIM BIT
        STA 2,0,3        ;STORE BACK IN RCB
        JMP @DSCHD       ;RETURN THRU SCHEDULER
ENQ2    MOVR 2,1         ;HOL RCB WITH CLAIM
        LDA 2,QUET1      ;PICK UP ADDR OF NEW TCB
        LDA 0,TPTY,2     ;PICK UP ITS PRIORITY
        MOVZL 0,0        ;GET TRUE PRIORITY
        MOV 1,2          ;AC2=C(RCB)
ENQ3    LDA 1,TPTY,2     ;PICK UP PRIORITY OF QUEUED TCB
        MOVZL 1,1        ;GET TRUE PRIORITY
        ADC 0,1          ;COMPARE PRIORITIES
```

```
        MOVZL 1,1,SNC   ;IS NEW TCB HIGHER
        JMP ENQ4        ;YES, INSERT HERE
        MOV 2,3         ;NO, AC3=A(QUEUED TCB)
        LDA 2,0,3       ;AC2=A(NEXT QUEUED TCB)
        MOV 2,2,SZR     ;ARE WE AT THE END OF THE STACK
        JMP ENQ3        ;NO, CONTINUE
ENQ4    LDA 1,QUET1     ;YES, GET ADDR OF NEW TCB
        MOVZL 2,2       ;SAVE POSS CLAIM BIT FOR RCB REST
        ADDR 1,1        ;HOLDING ITIN INSERT WORD
        MOVZR 2,2       ;READJUSTING LINK CHAIN ADDR
        STA @2,QUET1    ;STORE IT IN NEW TCB
        STA 1,0,3       ;STORE LINK INSERT WORD
        JMP @QSCHD      ;RETURN THRU SCHEDULER
```

```
;
;*******************************************
;
;       ENTRY POINT FOR  DEQU SYSTEM CALL
;       FORMAT:
;        DEQU
;       <ADDRESS>ADDRESS OF RCB
;       <RETURN>
;
;*******************************************
;
 DEQ.:  ISZ  SYS.       ;INDICATE SYSTEM MODE
        STA 2, AC2      ;SAVE IN AC2
        LDA 2,0,3       ;PICK UP RCB ADDRESS
        STA 2,QUET1     ;SAVE IT
        INC 3,3         ;ADJUST RETURN ADDR
        STA 3, RTN.     ;SAVE IT
        JSR @ENDEQ      ;CREATE A QUEUE BLOCK AND PLACE ITS
        JSR @ENDEQ+2    ;ADDR IN PENDING JOB STACK
        LDA @2,QUET1    ;PICK UP CONTENTS OF RCB
        SUB 1,1         ;INITIATE NEW RCB CONTENTS
        ADDOR 2,2,SNR   ;IS THERE A TASK QUEUED
        JMP DEQ1        ;NO, GO SAVE CLEARED RCB CONTENTS
        JSR @ENDEQ+2    ;YES, ENTER TOP QUEUE ENTRY IN STACK
        LDA 1,0,2       ;PICK UP CHAIN ADDR TO NEXT QUEUE
        ADDOR 1,1       ;INSERT RESOURCE CLAIM BIT AND
DEQ1    STA @1,QUET1    ;SAVE AS RCB CONTENTS
        JMP @QSCHD      ;EXIT THRU TASK SCHEDULER
;
QUET1   0               ;TEMP STORAGE
ENDEQ    QBLK           ;VECTOR TO TCB BUILDING ROUTINE
         SHED           ;VECTOR TO SCHED (AC2=A(PENDING TCB))
        JSPSH           ;VECTOR TO ROUTINE(PUSH TCB IN STAK)
```

```
;
;****************************************************
;
;       RI[illegible] STORAGE TABLES
;
;****************************************************
;
;        XMIT/ RCV CHANNEL STATUS TABLE
;
CST      BLK CHAN
;****************************************************
;
;        XMIT/ RCV CHANNEL MESSAGE TABLE
;
CMT      BLK CHAN
;
;****************************************************
;
;       STACK OF TCB ADDRESSES TO BE QUEUED
;
JSTK     .+1
        0
        BLK JOBS
;
;****************************************************
;
;       STACK OF TCB ADDRESSES IN PENDING QUEUE
;
QSTK    .+1
        0
        BLK JOBS
;
;****************************************************
;
;       ADDRESS STACK OF CLOCK QUEUE ENTRIES
;
CSTK    .+1
        0
        BLK JOBS
;
;****************************************************
;
;       CLOCK COUNT DOWN INTERVAL QUEUE
;
CLK      BLK 3*JOBS
;
;****************************************************
;
;       PENDING JOB TCB STORAGE AREA
;
JOB      BLK 7*JOBS
;
;****************************************************
```

```
;
;        SYSTEM MONITOR STACK
;
          IFN SMON
         177777    ;END OF BUFFER FLAG
BMON      BLK 2*SMON-1
EMON     0
         177777    ;END OF BUFFER FLAG
          ENDC

;*************************************************************

          END     ;END RTOS
```

```
;
;*****************************************************;
;
;      IIIIIII   N      N   IIIIIII   TTTTTTT        ;
;         I      NN     N      I         T           ;
;         I      N N    N      I         T           ;
;         I      N  N   N      I         T           ;
;         I      N   N  N      I         T           ;
;         I      N    N N      I         T           ;
;         I      N     NN      I         T           ;
;      IIIIIII   N      N   IIIIIII      T           ;
;
;*****************************************************;
;
;        PROJECT  GPS/DOPPLER HYBRID NAVIGATION SYSTEM
;
;        PROJECT ENGINEER  JACK GRAY
;
;            * INITIALIZATION PROGRAM "INIT" *
;
;SYSTEM DEFINITIONS
;
 DUSR JOBS=32
 DUSR CHAN=2      ;NUMBER OF  XMIT/ RCV CHANNELS
 DUSR TTYS=1
 DUSR FREQ=1
 DUSR SHALT=0     ;SYSTEM RESOURCES DEPLETED PARAMETER
                  ;0=> HALT; 1=> JUMP TO USER SUPPLIED
                  ;PROGRAM WITH ENTRY POINT ".SHLT"
 DUSR SMON=30     ;SYSTEM MONITOR
                                                       ;
;DEVICE/OPTION DEFINITIONS --
;       0 => NOT AVAILABLE
;       1 => DEVICE ON SYSTEM
                                                       ;
 DUSR   PWRFL=1 ;POWER FAIL MONITOR AUTO RESTART
 DUSR HSR=1
 DUSR HSP=1
 DUSR PRINT=1
 DUSR   PLOT=0  ;INCREMENTAL PLOTTER
 DUSR   CARD=0  ;CARD READER
 DUSR   DISK=0  ;DISK (FIXED HEAD)
 DUSR   A2D=0   ;ANALOG TO DIGITAL CONVERTER
 DUSR   TAPE=1  ;MAGNETIC TAPE UNIT
 DUSR   DCOM=0  ;DATA COMMUMNICATIONS MULTIPLEXER
 DUSR   QMUX=0  ;TYPE 4060 ASYNCHRONOUS MULTIPLEXOR
 DUSR   IBM=0   ;TYPE 4025 NOVA--SYSTEM 360 INTERFACE
 DUSR   DPACK=0         ;MOVING HEAD DISK HANDLER
 DUSR   SYNC=0  ;TYPE 4015 SYNCHRONOUS COMMUNICATIONS CONTROLLER
 DUSR JRB=0
 DUSR BUTN=0
 DUSR RCU=0
 DUSR SERIO=0
 DUSR FGATE=0
 DUSR HDMA1=1
 DUSR HDMA2=1
 DUSR   KW7S=1             ;NUMBER OF KW7'S
```

```
;*****************************************************************,
;                                                                ;
;    RTOS INITIALIZATION PROGRAM                                 ;
;                                          NEC00207  U           ;
;    THIS ROUTINE IS USED TO INITIALIZE THE REAL                 ;
;    TIME OPERATING SYSTEM.  IT IS USUALLY DESIREABLE            ;
;    TO LEAVE IT AVAILABLE IN CORE FOR DEBUGGING                 ;
;    PURPOSES.  ITS FUNCTION (PRIMARILY) IS TO                   ;
;    INITIALIZE STACKS AND CLEAR SWITCHES IN THE                 ;
;    SYSTEM                                                      ;
;                                                                ;
;*****************************************************************,

         TITL RTIN
         ENT INIT
         EXTN  SYS , CPTY, PMSK, QUE , RTC , OSTK, OPNT
         EXTN  CSTK, CPNT, JSTK, JPNT, CUCB, TTYO, BCHR
         EXTN  CLK , JOB , CST , START
         ENT    RINIT
         EXTN   INTP

         IFN SMON
         EXTD  EMON, MON
         ENDC

         NREL

         IFN IBM
         ENT  CTAB, STAB, TOP
CTAB     BLK 400          ;THE COMMAND AND STATUS TABLES
STAB     BLK 400          ;MUST LIE ON A MODULO 256 BOUNDARY
TOP
         ENDC

INIT     IORST
         INTDS
         LDFNW     1          ;GET RTOS INTERRUPT DISPATCHER ADDRESS
                   INTP
         STA       1,1        ;MAKE IT ACTIVE
RINIT    IORST    ;ENTER HERE UPON RE-INITIALIZATION
         INTDS
         LDA 3,TOP1        ;LOCATIONS TO BE ZEROED
         SUB 0,0
INO      LDA 2,0,3
         MOV 2,2,SNR       ;CHECK IF END OF TABLE
         JMP IN1
         MOVL# 2,2,SNC     ;SKIP IF HANDLER NOT LOADED
         STA 0,0,2
         INC 3,3
         JMP INO

IN1      LDA 3,XRTAB       ;CLEAR  XMIT/ RCV CHANNELS
         LDA 2,COUNT+1
         NEGZL 2,2
         STA 0,0,3
         INC 3,3
         INC 2,2,SZR
         JMP  -3
```

```
IN2     LDA 0,C7         ;INITIALIZE TCB BLOCK STACK
        LDA 1,COUNT
        NEG 1,1
        LDA 2,STAK1+1
        LDA 3,@STAK1+2
        INC 3,3
        STA 2,0,3
        ADD 0,2
        INC 1,1,SZR
        JMP  -4
        STA 3,@STAK1

IN3     LDA 0,C3         ;INITIALIZE CLOCK BLOCK STACK
        LDA 1,COUNT
        NEG 1,1
        LDA 2,STAK2+1
        LDA 3,@STAK2+2
        INC 3,3
        STA 2,0,3
        ADD 0,2
        INC 1,1,SZR
        JMP  -4
        STA 3,@STAK2

IN4     ADC 1,1                  ;SET AC1 NEGATIVE
        STA 1,@BRQST     ;SET BREAK REQUEST INACTIVE

IN5     LDA 3,@STAK3+1   ;INITIALIZE JOB STACK
        STA 3,@STAK3

         IFN IBM
IN6     JSR @IBMIN       ;INITIALIZE TYPE 4025 INTERFACE
         ENDC

         IFN QMUX
IN7     JSR @QMUXIN      ;INITIALIZE TYPE 4060 MULTIPLEXOR
         ENDC

         IFN SYNC
IN8     JSR @.T4015      ;INITIALIZE TYPE 4015 CONTROLLER
         ENDC

         IFN DPACK
IN9     JSR @.DPACK      ;INITIALIZE MOVING HEAD DISK HANDLER
         ENDC

         IFN SMON
        LDA 0, EMON      ;GET START ADDRESS OF MONITOR
        STA 0, MON       ;SET MONITOR POINTER
         ENDC

        INTEN            ;TURN INTERRUPT FACILITY ON
        JMP @ +1         ;BRANCH TO USER TASK AT START
        START            ;TO BE DEFINED EXTERNALLY

TOP1     +1
         SYS             ;SYSTEM OPERATING MODE
         CPTY            ;TASK PRIORITY STORAGE
         PMSK            ;HARDWARE INTERRUPT MASK
```

```
QUE                                                     ;
RTC             ;REAL TIME CLOCK ACTIVE SWITCH
CUCB                                                    ;
TTY0            ;ADDRESS OF TTY UNIT 0 DUCB
IFE TTYS-2                                              ;
EXTN  TTY1      ;ADDRESS OF TTY UNIT 1 DUCB
TTY1                                                    ;
ENDC                                                    ;
                                                        ;
IFE TTYS-3                                              ;
EXTN  TTY1      ;ADDRESS OF TTY UNIT 1 DUCB
TTY1                                                    ;
EXTN  TTY2      ;ADDRESS OF TTY UNIT 2 DUCB
TTY2                                                    ;
ENDC                                                    ;
                                                        ;
IFN HSP         ;HIGH SPEED PAPER TAPE PUNCH
EXTN  PTP1                                              ;
PTP1                                                    ;
ENDC                                                    ;
                                                        ;
IFN HSR         ;HIGH SPEED PAPER TAPE READER
EXTN  PTR1                                              ;
PTR1                                                    ;
ENDC                                                    ;
                                                        ;
IFN PRINT       ;LINE PRINTER
EXTN  LPT1                                              ;
LPT1                                                    ;
ENDC                                                    ;
                                                        ;
IFN PLOT        ;INCREMENTAL PLOTTER
EXTN  PLT1                                              ;
PLT1                                                    ;
ENDC                                                    ;
                                                        ;
IFN A2D         ;ANALOG TO DIGITAL CONVERTER
EXTN  ADC1                                              ;
ADC1                                                    ;
ENDC                                                    ;
                                                        ;
IFN CARD        ;CARD READER
EXTN  CDR1                                              ;
CDR1                                                    ;
ENDC                                                    ;
                                                        ;
IFN DISK        ;FIXED HEAD DISK
EXTN  DSK1                                              ;
DSK1                                                    ;
ENDC                                                    ;
                                                        ;
IFN TAPE        ;MAGNETIC TAPE
EXTN  MTA1                                              ;
MTA1                                                    ;
ENDC                                                    ;
                                                        ;
IFN DCOM        ;DATA COMMUNICATIONS MULTIPLEXER
```

```
EXTN   DCM1
DCM1
ENDC

IFN URB          ;URB UNIT NUMBER 0
EXTN   URB1
URB1
ENDC

IFN URB-1        ;URB UNIT NUMBER 1
EXTN   URB2
URB2
ENDC

IFN BUTN         ;PUSH BUTTON BOARD #1
EXTN   BUT1
BUT1
ENDC

IFN BUTN-1       ;PUSH BUTTON BOARD #2
EXTN   BUT2
BUT2
ENDC

IFN      RCU
EXTN     RCU0
RCU0
ENDC

IFN      RCU-1
EXTN     RCU1
RCU1
ENDC

IFN      SERIO
EXTN     SER0
SER0
ENDC

IFN      SERIO-1
EXTN     SER1
SER1
ENDC
IFN      KW7S
EXTN     KUCB
KUCB
         ENDC
IFN HDMA1
EXTN   DM10
DM10
ENDC
IFN HDMA2
EXTN   DM20
DM20
ENDC
```

```
        0                  ;TABLE END INDICATOR

BRCST    [illegible]CHR    ;B[illegible] CHAR  STORAGE

COUNT   JOBS               ;NUMBER OF TASKS ALLOWED
        CHAN               ;NUMBER OF  XMIT/ RCV CHANNELS

XRTAB    CST               ; XMIT/ RCV CHANNEL ACTIVE TABLE

C7      7
C3      3

STAK1    QPNT              ;JOB STACK VARIABLES
         JOB
         OSTK

STAK2    CPNT              ;CLOCK STACK VARIABLES
         CLK.
         CSTK

STAK3:   JPNT              ;STACK POINTERS
         JSTK

        .IFN DPACK
         EXTN  DKPO
 DPACK:        .DKPO               ;MOVING HEAD DISK INITIALIZATION SUBR.
         ENDC

         IFN IBM
         EXTN .IBMO
IBMIN    IBMO              ;TYPE 4025 INITIALIZATION SUBROUTINE
         ENDC

        .IFN QMUX
        .EXTN .QMXO
QMUXIN:        .QMXO               ;TYPE 4060 MUX. INITIALIZATION SUBR.
         ENDC

         IFN SYNC
        .EXTN .TSNO
 T4015. .TSNO              ;TYPE 4015 INITIALIZATION SUBROUTINE
         ENDC

         END INIT
```

```
;
;****************************************;
;                                        ;
;    *****     *****     *****           ;
;    *         *   *     *               ;
;    * ***     *****     *****           ;
;    *   *     *             *           ;
;    *****     *         *****           ;
;                                        ;
;****************************************;
;
;        * GPS SYSTEM I/O PROGRAM *
;
;        GPS DATA IS ACCESSED THROUGH A 16 BIT PARALLEL I/O
;
;
;        MEMORY ACCESS (DMA). IT IS CONTROLLED BY THE EXECUTIVE'S
;
;        INTERRUPT SYSTEM.
;
;
 TITL JO SR
 NREL
 ENT  DMA1,  DM1Q
 EXTN  QUIT, IOEND, DISN
 EXTD .SERV, STAK
;
;ADDRESS OF DEVICE INITIALIZATION ROUTINE
;
GOPR: IOXIN
;
;INTERRUPT SERVICE ROUTINE
;
 DMA1:   JSR @.SERV
         177777
         0
         0
         0
         0
         0
         0
         0
GBADR:   GBLK
;
;
LDA 3,GTCBA,2
MOV 3,3,SNR
JMP ERR-1
JMP ITERM
;
;DEVICE INITIALIZATION ROUTINE
;
IOXIN:   LDA 3,GBADR
         ISZ GQBSY,3
         LDA 1,GTCBA,3
         MOV 1,1,SZR
         JMP @.STAK
```

```
;
NEXT     STA 2,GTCBA,3
         STA 1,GQBSY,3
         STA 0,GAD,3
         LDA 2,TAC3,2

         LDA 1,IERTN,2
         STA 1,GERTN,3
         LDA 1,IDPTR,2
         STA 1,GDADR,3
         STA 1,1,3
         LDA 1,IGCNT,2
         STA 1,GCNT,3
         LDA 1,GCNT,3
         NEG 1,1
         STA 1,GCNT,3
;
NIOC DCC2
;
         LDA 0,GDADR,3
         DOA 0,DCC2
         LDA 0,GCNT,3
         DOBS 0,DCC2
          QUIT
;
         NIOS DCC2
ERR:     LDA 2,DIDBO,2
         JMP @.+1
          DISN
;
IBAD:    LDA 3,GTCBA,2
         LDA 1,GERTN,2
         STA 1,TPC,3
;
ITERM:   LDA 3,GTCBA,2
         STA 0,TAC3,3
         LDA 0,DIDBO,2
         JMP @.+1
          IOEND
;
GBLK:    0
         0
         JMP 0,3
         0
         DCC2
         0
         0
         0
         0
         0
         NEXT
          DMA1+3
         0
         0
         0
         0
```

```
 DM10=GBLK
,
GTCBA=0
DCC2=7
GDADR=5
GCNT=6
GQBSY=7
DIDBO=13
GERTN=16
IDPTR=2
IGCNT=3
IERTN=4
TAC3=5
TPC=6
GAD=14
```

```
;
;**********************************************;
;                                              ;
;  DD    O   PP   PP   L     EEE  RR           ;
;  D D  O O  P P  P P  L     E    R R          ;
;  D D  O O  PP   PP   L     EEE  RR           ;
;  D D  O O  P    P    L     E    R R          ;
;  DD    O   P    P    LLL   EEE  R R          ;
;                                              ;
;**********************************************;
;
;                * AN/ASN-128 DOPPLER NAVIGATION SYSTEM I/O PROGRAM *
;
;
;                DOPPLER DATA IS ACCESSED THROUGH THE ARINC INTERFACE
;
;                AND LOADED VIA 256 WORD BLOCKS THROUGH DIRECT MEMORY
;
;                ACCESS (DMA)  IT IS CONTROLLED BY THE EXECUTIVE S
;
;                INTERRUPT SYSTEM
;
 TITL PATTI
 NREL
 ENT  DMA2, DM20
 EXTN  QUIT, IOEND, DISN
 EXTD  SERV, STAK
;
;ADDRESS OF DEVICE INITIALIZATION ROUTINE
;
DOPR: IOXIN
;
;INTERRUPT SERVICE ROUTINE
;
 DMA2:  JSR @.SERV
        177777
        0
        0
        0
        0
        0
        0
        0
DBADR:  DPBLK
;
;
LDA 3,DTCBA,2
MOV 3,3,SNR
JMP FOUL-1
JMP ITERM
;
;DEVICE INITIALIZATION ROUTINE
;
IOXIN: LDA 3,DBADR
       ISZ DQBSY,3        ;SET QUEUE BUSY
       LDA 1,DTCBA,3      ;GET LINK WORD
       MOV 1,1,SZR        ;IS DEVICE BUSY
       JMP @.STAK         ;YES
;
```

```
NEXT     STA 2,DTCBA,3
         STA 1,DQBSY,3
         STA 0,SAD,3
         LDA 1,[illegible],2
         LDA 1,IERTN,2
         STA 1,DERTN,3
         LDA 1,IDPTR,2
         STA 1,DDADR,3
         STA 1,1,3
         LDA 1,IDCNT,2
         STA 1,DCNT,3
LDA 1,DCNT,3
         NEG 1,1          ;NEGATE DATA COUNT
         STA 1,DCNT,3
;
NIOC DCC1
;
LDA 0,DDADR,3
DOA 0,DCC1
LDA 0,DCNT,3
DOB 0,DCC1
LDA 0,SAD,3
DOC 0,DCC1
NIOS DCC1
          QUIT
;
         NIOC DCC1
FOUL:  LDA 2,DIDBO,2
         JMP @.+1
          DISN
;
IBAD     LDA 3,DTCBA,2
         LDA 1,DERTN,2
         STA 1,TPC,3
;
ITERM:  LDA 3,DTCBA,2
         STA 0,TAC3,3
         LDA 0,DIDBO,2
         JMP @.+1
          IOEND
;
DPBLK    0         ;DTCBA
         0         ;
         JMP 0,3   ;
         0         ;
         DCC1      ;DVCDE
         0         ;DDADR (DATA ADDRESS)
         0         ;DCNT (DATA COUNT)
         0         ;DQBSY
         0         ;
         0         ;
```

```
         NEXT     ; DNIOR (NEXT I/O ADDRESS)
          DMA2+3  , DIDBO (ADDRESS OF INTERRUPT DATA)
         0        ,
         0        ,
         0        , DERTN (ERROR RETURN)
         0        . DMODE

   [illegible]

[illegible]9A=[illegible]
DVCDE=4
DCC1=46
DDADR=5
DCNT=6
DDBSY=7
DIDBO=13
DNIOR=12
DERTN=16
 IDPTR=2
 IDCNT=3
 IERTN=4
 TAC3=5
 TPC=6
 SAD=14
          END
```

```
;PARAMETERS FOR COMMON BLOCK PT (POINT)
;LENGTH OF PT = 17
        DUSR    ISP     =0
        DUSR    IZN     =ISP+1
        DUSR    KZL     =IZN+1
        DUSR    INZN    =KZL+1
        DUSR    XLAT    =INZN+1
        DUSR    XLON    =XLAT+2
        DUSR    XN      =XLON+2
        DUSR    E       =XN+2
        DUSR    HTMGD   =E+2
        DUSR    GE      =HTMGD+1
        DUSR    GN      =GE+2


        TITL    FORMT
;**********************************************************
;
;       F O R M A T T E R       R O U T I N E S
;
;**********************************************************
;
;
;
;    THIS SOFTWARE PACKAGE CONTAINS MANY SUBROUTINES USEFUL
;    IN DECODING / PACKING OF ASCII INPUT / OUTPUT BUFFERS
;
;
;     AS A SYSTEM RESOURCE, THE FORMATTER MUST BE FIRST
;     "LOCKED" BY THE USER WITH:
;
;
;               ENQ
;               FMLOK
;               <RETURN>
;
;
;     THEN THE USER MAY CALL ANY ROUTINE IN THE PACKAGE
;
;
;
;     WHEN THE USER IS FINISHED WITH THE PACKAGE, HE MUST
;     RELEASE IT WITH:
;
;
;               DQU
;               FMLOK
;               <RETURN>
;
;
;
;     EACH ROUTINE USED MUST BE DECLARED AS AN EXTERNAL
;     NORMAL ( EXTN) IN THE CALLING PROGRAM TO SET UP LINKAGE
;     ADDRESSES.  ALSO, REGISTERS ARE NOT SAVED ON RETURN.
;
;
```

```
;
;      THE INPUT AND OUTPUT FORMATTING ROUTINES AND THEIR
;     CALLING SEQUENCES ARE GIVEN IN THE FOLLOWING LISTINGS
;

          ENT     FORMT
          ENT     I.SET,I.BYT,GBYTE,I.BAC,I.COM,I.LOC,I.INT,I.PTR
          ENT     I.FLT,O.SET,O.BYT,O.INT,O.FLT,O.MOV
          ENT     O.PAD,O.LOC
          ENT     LLMODE,SPHERE
          ENT     GNXT,INBND,BYTA
          ENT      ISET, IBYT, GBYTE, IBAC, ICOM, ILOC, IINT, IPTR
          ENT      IFLT, GNXT, INBND, OSET, OBYT, OINT, OFLT, OMOV, OPAD
          ENT      OLOC,OBUF
          EXTN     ENQU, DEQU,FMLOF
          EXTD     MEMR, MG2UM, UM2GP, UM2MG, GP2UM
          NREL
          COMM    PT 17
          ZREL
 ISET:   I.SET
 IBYT:   I.BYT
 GBYTE:  GBYTE
 IBAC:   I.BAC
 ICOM:   I.COM
 ILOC:   I.LOC
 IINT:   I.INT
 IPTR:   I.PTR
 IFLT:   I.FLT
 GNXT:   GNXT
 INBND:  INBND
 OSET:   O.SET
 OBYT:   O.BYT
 OINT:   O.INT
 OFLT:   O.FLT
 OMOV:   O.MOV
 OPAD:   O.PAD
 OLOC:   O.LOC
SPHERE:  0
LLMODE:  0
          NREL


;*************************************************************
;
;    SUBROUTINE I.SET -  THIS ROUTINE SETS UP THE BYTE POINTER
;    TO BE USED IN THE PROCESSING OF INPUT BUFFERS.  IT MUST BE
;    CALLED BEFORE ANY INPUT FORMATTING ROUTINE.
;    THE LAST BYTE OF THE INPUT BUFFER MUST BE A BINARY ZERO
;
;    AC0 = STARTING BYTE ADDRESS OF BUFFER
;    JSR @I.SET
;
I.SET:   STA      0,BYTA      ; STORE BYTE POINTER
         SUB      1,1         ; RESET THE RUNNING
         STA      1,RUNB      ;    BYTE COUNT
         JMP      0,3         ; RETURN
```

```
        FORMT   = I SET
;***********************************************************
;
;   SUBROUTINE I PTR -  THIS ROUTINE WILL UPDATE THE INPUT BYTE
;   POINTER OF IN AN INPUT CONTROL BLOCK TO THE CURRENT VALUE.
;
;   AC2 = INPUT CONTROL BLOCK ADDRESS
;   JSR @I.PTR
;
I PTR   LDA     0,BYTA       ; GET THE CURRENT BYTE POINTER
        STA     0,1,2        ; STORE IT IN THE CONTROL BLOCK
        JMP     0,3


;***********************************************************
;
;   SUBROUTINE GBYTE -  THIS IS USED INTERNALLY TO GET THE
;   NEXT BYTE IN THE INPUT BUFFER.
;
;   JSR  GBYTE
;   <END OF LINE RETURN>
;   <NORMAL RETURN>
;
GNXT    SUB     1,1          ; GENERATE A 0
        JMP     .+2          ; GET PAST NEXT INSTRUCTION
GBYTE:  SUBZL   1,1          ; GENERATE A 1
        LDA     2,BYTA       ; LOAD BYTE POINTER
        MOVZR   2,2          ; SHIFT FOR WORD ADDR
        LDA     0,0,2        ; GET BUFFER WORD
        MOV     0,0,SZC      ; WHICH SIDE?
        MOVS    0,0          ; SWAP FOR LEFT
        ANFNW   0            ; MASK OUT
                177          ; 7 BIT MASK
        MOV     0,0,SNR      ; IS THE BYTE ZERO (END-OF-LINE)?
        JMP     0,3          ; EOL - EXIT
        ISZ     BYTA         ; BUMP ADDR
        ISZ     RUNB         ; BUMP COUNT
        MOV     1,1,SNR      ; IS IT A GBYTE ENTRY
        JMP     1,3          ; NO,  NORMAL RETURN
        LDFNW   1            ; LOAD ASCII
                "            ; BLANK
        SUB#    1,0,SNR      ; IS THE BUTE A BLANK?
        JMP     GBYTE        ; YES - GET ANOTHER ONE
        JMP     1,3          ; NORMAL RETURN


;***********************************************************
;
;   SUBROUTINE I BAC -  THIS ROUTINE BACKS UP THE INPUT
;   BUFFER BY ONE BYTE.
;
;   JSR  I BAC
```

```
I BAC    LDA     0,RUNB       ; GET RUNNING BYTE COUNT
         MOV     0,0,SNR      ; ALREADY ZERO?
         JMP     0,3          ; YES - RETURN
         DSZ     RUNB         ; DECREMENT
         NOP                  ;   RUNNING BYTE COUNT
         DSZ     BYTA         ;   AND
         NOP                  ;   BYTE ADDRESS
         JMP     0,3          ; RETURN

         I BYT=GBYTE

; ***************************************************************
;
;   SUBROUTINE I COM -  THIS ROUTINE SCANS TO THE NEXT COMMA
;   IN THE INPUT BUFFER
;
;   JSR   I COM
;
I COM    STA     3,IRT2       ; STORE RTA
         JSR     GBYTE        ; GET BYTE
         JMP@    IRT2         ; EOL RETURN
         LDA     1,COMMA      ; GET ASCII FOR COMMA
         SUB     0,1,SZR      ; IS THE BYTE A COMMA?
         JMP     I COM+1      ; NO  - GET ANOTHER BYTE
         ISZ     IRT2         ; YES - RETURN
         JMP@    IRT2
IRT2      BLK    1
RUNB      BLK    1
BYTA      BLK    1

; ***************************************************************
;
;   SBUROUTINE I INT -  THIS ROUTINE RETURNS THE NEXT INTEGER
;   VALUE FROM THE INPUT BUFFER.  IF END OF LINE WAS
;   REACHED, THE EOL RETURN IS TAKEN WITH THE VALUE UP TO THE
;   EOL IN AC0.  IF AN INVALID CHARACTER WAS ENCOUNTERED, THE
;  VALUE UP TO THE INVALID CHARACTER IS RETURNED IN AC0 AND
;   AC1 HAS THE INVALID CHARACTER.
;
;   JSR   I INT
;   (END-OF-LINE RETURN)
;   (INVALID INPUT ...)
;   (NORMAL RETURN)        AC0 = NEXT INTEGER FIELD
;
                              ; STORE RETURN
                              ; GET NEXT FIELD AS FLOATING
                              ; EOL - RETURN
                              ; INVALID INPUT
                 IRT2         ; BUMP RTA
                 IRT2         ; BUMP RTA
         FIX                  ; CONVERT TO INTEGER
         MOV                  ; MOVE THE RESULT
                              ; NORMAL RETURN
```

```
*******                                    ********************************

          [illegible] ROUTINE RE[illegible] THE NEXT FLOATING
          [illegible] INPUT BUFFER [illegible] OR INVALID
          THE [illegible] ACCUMULATED
          POINT [illegible]

          [illegible] A FLOATING [illegible]

        STA     [illegible]        ; [illegible]RE RETURN ADDR
        SUB     [illegible]        ; CLEAR AC0
        SUB     [illegible]        ;   AC1
        DST     ANS              ; RESET THE
                                 ;   RUNNING [illegible]
        DST     FR               ; RESET THE
                                 ;   FRACTIONAL ACCUM
        DLD     F1               ; LOAD FLOATING
                                 ;   POINT ONE
        DST     FRMUL            ; STORE TO FRACTIONAL
                                 ;   MULTIPLIER
        STA     1,DFLG           ; RESET DECIMAL POINT FLAG
        STA     1,SFLG           ; RESET SIGN FLAG
        JSR     GBYTE            ; GET FIRST BYTE
        JMP@    IRT3             ; END-OF-LINE EXIT
        LDA     1,MINUS          ; IS IT A
        SUB     0,1,SZR          ;   MINUS SIGN?
        JMP     NXTC+2           ; PROCESS CHARACTER
        ISZ     SFLG             ; BUMP SIGN FLG
NXTC    JSR     GBYTE            ; GET NEXT BYTE
        JMP     EOFX
        LDA     1,DECPT          ; IS IT A
        SUB     0,1,SNR          ;   DECIMAL POINT?
        JMP     DPNT             ; YES - SET FLAG
        LDA     1,COMMA          ; IS IT A
        SUB     0,1,SNR          ;   COMMA?
        JMP     EOFX             ; YES - END OF FIELD
; TEST FOR NUMERIC INPUT
        LDA     1,B60            ; SET LOWER LIMIT
        LDA     2,B71            ; SET UPPER LIMIT
        JSR     INBND            ; TEST IF BYTE (AC0) IN BOUNDS
        JMP     ERRX             ; NO - ERROR EXIT
        SUB     1,0              ; FORM LEGAL DIGIT
        MOV     0,1              ; GET FLOATING
        SUB     0,0              ;   POINT
        FLD                      ;   REPRESENTATION
        FNM
        DST     DIGIT            ; STORE
        LDA     2,DFLG           ; DECIMAL POINT
        MOV     2,2,SZR          ;   SET?
        JMP     FPART            ; YES - DO FRACTION PART
```

```
;       INTEGER PART
        DLD             ; LOAD
                ANS     ;   CURRENT ANSWER
        FMPNM           ; MULTIPLY BY
                FD10    ;   10D
        FADNM           ; ADD ON
                DIGIT   ;   CURRENT DIGIT
        DST             ; STORE ANSWER
                ANS
        JMP     NXTC    ; GET NEXT CHAR
;       FRACTION PART
FPART   DLD             ; LOAD
                FRMUL   ;   FRACTIONAL MULTIPLIER
        FDVNM           ; DIVIDE
                FD10    ;   BY 10D
        DST             ; STORE THE RESULT
                FRMUL
        FMPNM           ; MULTIPLY WITH
                DIGIT   ;   CURRENT DIGIT
        FADNM           ; ADD RESULT TO
                FR      ;   FRACTION PART
        DST             ; STORE THE
                FR      ;   RESULT
        JMP     NXTC    ; GET NXTCHAR
DPNT    LDA     1,DFLG  ; IS FLAG
        MOV     1,1,SZR ;   ALREADY SET?
        JMP     ERRX    ; YES - ERROR EXIT
        ISZ     DFLG    ; NO - SET FLAG
        JMP     NXTC    ; GET NEXT CHAR
ERRX:   DSZ     IRT3    ; INVALID INPUT ERROR
        MOV     0,2     ; SAVE THE INVALID CHARACTER
;       NORMAL EXIT
EOFX:   DLD             ; LOAD
                ANS     ;   INTEGER PART
        FADNM           ; ADD
                FR      ;   FRACTION PART
        LDA     3,SFLG  ; IS MINUS SIGN
        MOV     3,3,SZR ;   FLAG SET?
        FNG             ; YES - NEGATE THE RESULT
        LDA     3,IRT3
        JMP     2,3

ANS:     BLK    2
FR:      BLK    2
DIGIT:   BLK    2
FRMUL:   BLK    2
SFLG:    BLK    1
DFLG:    BLK    1
MINUS:  "-
DECPT:  56
FD10:   10.0
F1:     1.0
B71:    71
B60:    60
IRT3:    BLK    1
COMMA:  ",
```

```
;**************************************************************
;
;   SUBROUTINE GET2 -  THIS IS AN INTERNAL ROUTINE TO GET TWO
;   CONSECUTIVE ASCII BYTES AND PACK THEM INTO ONE WORD
;
;   JSR  GET2  -  INTERNAL ROUTINE   IMMEDIATELY
;                   EXITS ON END-OF-LINE AND
;                   INVALID INPUT
;                   CALLED BY I UTM
GET2:    STA    3,IRT5      ; STORE RETURN ADDR
         SUB    0,0         ; RESET
         STA    0,PCHR      ;   VARIABLE
         JSR@   GBYTE       ; GET A BYTE
         JMP    INIX        ; EOL - RETURN
         LDA    1,B60       ; GET LOWER LIMIT
         LDA    2,B71       ; GET UPPER LIMIT
         JSR    INBND       ; CHECK IF BYTE IS NUMERIC
         JMP    INIX        ; NO - INVALID INPUT EXIT
         LDA    1,PCHR      ; FIRST BYTE ALREADY
         MOVS   1,1,SZR     ;   SET?
         JMP    +3          ; YES - ADD THE OTHER
         STA    0,PCHR      ; NO - STORE ONE
         JMP    GET2+3      ;   GET ANOTHER
         ADDS   1,0         ; PACK THE SECOND BYTE
         JMP@   IRT5        ; RETURN TO I UTM
IRT5:    .BLK   1
PCHR:    .BLK   1

;**************************************************************
;
;   SUBROUTINE INBND -  THIS IS AN INTERNAL ROUTINE TO TEST
;   WHETHER A NUMBER IS WITHIN THE LIMITS OF TWO OTHERS.
;
;   AC0 = BYTE
;   AC1 = LOWER LIMIT
;   AC2 = UPPER LIMIT
;   JSR  INBND
;   <ERROR RETURN>
;   <IN BOUNDS RETURN>
;
INBND:   SUBL#  1,0,SNC     ; COMPARE LOWER LIMIT
         SUBL#  0,2,SZC     ; COMPARE UPPER LIMIT
         JMP    0,3         ; ERROR
         JMP    1,3         ; SUCCESS RETURN
```

```
        . TITL MATH

******************************************************************************
*                                                                            *
*                                                                            *
*        MATH ROUTINES                                                       *
*                                                                            *
*                                                                            *
*                                                                            *
******************************************************************************

 THIS SOFTWARE PACKAGE CONTAINS MANY SUBROUTINES USEFUL IN SOLVING
 TRIGONOMETRIC EQUATIONS


                                                                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       A R C S I N   C A L C U L A T I O N
;
;       ENTRY:  AC0,AC1 CONTAIN INPUT PARAMETER
;
;       EXIT:   AC0,AC1 CONTAIN THE ARCSIN
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ASIN:   STA     3,ASRET         ;SAVE RETURN ADDRESS
        DST                     ;SAVE THE DATA
                TEMP3                                                       ;
        FMPNM                   ;FORM X**2
                TEMP3                                                       ;
        DST                     ;SAVE IT
                TEMP4                                                       ;
        DLD                     ;FLOATING POINT 1
                F1                                                          ;
        FSBNM                   ;FORM 1-X**2
                TEMP4                                                       ;
        JSR@    PSQRT           ;SQUARE ROOT
        DST                     ;SAVE
                TEMP4                                                       ;
        MOV     0,0,SNR         ;SKIP IF SQRT .NE. 0
        JMP     OVF             ;OVERFLOW
        DLD                     ;GET X AGAIN
                TEMP3                                                       ;
        FDVNM                   ;FORM X/SQRT(1-X**2)
                TEMP4                                                       ;
        JSR@    PATAN           ;ARCTANGENT
        JMP@    ASRET           ;RETURN
                                                                            ;
OVF:    DLD                     ;LOAD PIE/2
                PIE2                                                        ;
        LDA     2,TEMP3         ; MS 16 BITS OF X
        MOVL#   2,2,SZC         ;SKIP IF X WAS .GE. 0
        FNG                     ;NEGATE
        JMP@    ASRET           ;RETURN

PSQRT:  SQRT                    ;POINTER TO SQRT
PATAN:  ATAN
ASRET:  0                       ;RETURN ADDRESS
TEMP3:  0.0
TEMP4:  0.0
```

```
;*************************************************************
;
;   SUBROUTINE O.SET -  THIS ROUTINE SETS UP THE OUTPUT BUFFER
;   BYTE POINTER AND MUST BE CALLED BEFORE ANY OTHER OUTPUT
;   FORMATTING ROUTINE.  ALSO THE CALLER MUST BE CAREFUL NOT
;   PACK BEYOND THE BOUNDS OF HIS BUFFER.
;
;   AC0 = BUFFER STARTING BYTE ADDRESS
;   JSR  O.SET
;
O.SET:   STA     0,OBUF       ; STORE BYTE POINTER
         JMP     0,3          ; RETURN
OBUF:    .BLK    1

;*************************************************************
;
;   SUBROUTINE O.BYT -  THIS ROUTINE STORES A BYTE PASSED AS
;   ARGUMENT INTO THE OUTPUT BUFFER.  THE BYTE POINTER IS
;   INCREMENTED AFTER A STORE.
;
;   AC0 = OUTPUT BYTE
;   JSR  O.BYT
;
O.BYT:   ANFNW   0            ; ISOLATE LOW-ORDER
                 377          ;   8 BITS
         LDA     2,OBUF       ; GET BUFFER BYTE PTR
         MOVZR   2,2          ;GET WORD ADDRESS
         LDA     1,0,2        ;GET THE BUFFER WORD
         MOV     1,1,SNC      ;SIGN BIT SET FROM BYTE ADDRESS?
         JMP     RSIDE        ;STORE ON RIGHT
         ANFNW   1            ;MASK FOR RIGHT
                 377
         MOVS    0,0          ;SWAP THE CHARACTER TO BE STORED
         JMP     STWRD        ;GO STORE THE WORD
RSIDE:   ANFNW   1            ;MASK FOR LEFT CHARACTER
                 177400
STWRD:   ADD     1,0          ;ADDIN THE BYTE
         STA     0,0,2        ; STORE IN OUTPUT BUFFER
         ISZ     OBUF         ; BUMP BUFFER BYTE POINTER
         JMP     0,3          ; RETURN


;*************************************************************
;
;   SUBROUTINE O.INT -  THIS ROUTINE FORMATS A SINGLE PRECISION
;   INTEGER INTO A USER SPECIFIED FIELD WIDTH.  THE INTEGER IS
;   PACKED FROM THE CURRENT BYTE POINTER POSITION, RIGHT
;   JUSTIFIED WITH BLANK FILL TO THE LEFT.  IF THERE IS NOT
;   ENOUGH ROOM FOR THE INTEGER, DOLLAR SIGNS ARE PADDED INTO
;   THE FIELD.
;
;   AC0 = INTEGER
;   JSR  O.INT
;   <FIELD WIDTH>
;
```

```
O.INT   STA     3,ORT2      ; STORE RETURN
        MOV     0,1
        SUBZL   0,0         ; ACO <= 1
        SMPY    0           ; CONVERT TO DOUBLD PRECISION
        FLO                 ; CONVERT TO F.P.
        FNM
        LDA     3,ORT2      ; RELOAD RTA
        LDA     2,0,3       ; PICK UP FIELD WIDTH
        STA     2,.+2       ; STORE
        JSR     O.FLT       ; CALL FLOATING FORMATTER
         BLK    1           ;   INTEGER FIELD WIDTH
        0                   ;   DECIMAL FIELD WIDTH
        LDA     3,ORT2
        JMP     1,3
ORT2     BLK    1
TMP2     BLK    1


;***********************************************************
;
;   SUBROUTINE O.FLT -  THIS ROUTINE FORMATS A FLOATING POINT
;   VALUE INTO THE OUTPUT BUFFER.  THE NUMBER IS RIGHT JUSTIFIED
;   INTO THE SPECIFIED FIELD WITH BLANK FILL TO THE LEFT.  IF THE
;   INTEGER PART DOES NOT FIT, THE DECIMAL FIELD SIZE IS
;   DECREASED UNTIL THE INTEGER PART  HAS ENOUGH ROOM.  IF
;   THE TOTAL FIELD WIDTH IS TOO SMALL, DOLLAR SIGNS ARE PADDED
;   INTO THE OUTPUT BUFFER.
;
;   ACO,AC1 = FLOATING VALUE
;   JSR  O.FLT
;   <INTEGER FIELD WIDTH>
;   <DECIMAL FIELD WIDTH>
;   <RETURN>
;
;    EXAMPLE:    VALUE = 25.560
;                JSR  O.FLT
;                5
;                4
;   IS IN EFFECT A FORMAT SPECIFICATION OF F9.4  IN FORTRAN
;   TERMS.  THE RESULT IS:        BBB25.5600
;   WHERE "B" STANDS FOR BLANK.
;


O.FLT:  STA     3,ORT3      ; STORE RETURN
        LDFNW   2           ; LOAD ASCII
BLANK:          40          ;   BLANK
        MOVL#   0,0,SNC     ; IS NUMBER NEGATIVE?
        JMP     OF1         ; NO
        FNG                 ; NEGATE THE NUMBER
        LDFNW   2           ; LOAD ASCII
```

```
MSIGN              "-         ;   MINUS SIGN
OF1     STA        2,OSIGN    ; STORE SIGN CHARACTER
        SUB        2,2        ; SET FORMATTING FLAGS
        STA        2,FIRST
        INC        2,2
        STA        2,FIELD
        LDA        [illegible] ; RELOAD RTA
        LDA        [illegible] ; PICK UP INTEGER FIELD
        STA        [illegible] ; SAVE IT
        LDA        3,1,3      ; PICK UP DECIMAL FIELD WIDTH
        STA        3,NDEC     ; SAVE IT
        MOV        3,3,SZR    ; IS IT ZERO?
        INC        2,2        ; NO - ADD A PLACE FOR THE DECIMAL
        ADD        3,2        ; GET TOTAL REQ'D FIELD WIDTH
        STA        2,FW
;    DETERMINE FUDGE FACTOR
        LDA        2,FDMAX    ; GET MAP SIZE OF 'FUDGE' TABLE
        SUBL#      3,2,SZC    ; IS DECIMAL FIELD WIDTH BIFFER?
        JMP        OF2        ; YES - FORGET IT
        ADD        3,3        ; DOUBLE THE INDEX
        ADFNW      3          ; ADD ON FACTOR TABLE BASE ADDR
                   FFTAB
        STA        3,.+2      ; STORE THE ADDRESS
        FADNM                 ; ADD ON THE FUDGE FACTOR
                   .BLK   1   ; STORAGE FOR THE ADDRESS
;    DETERMINE NECESSARY WIDTH OF OUTPUT VALUE
OF2     FDVNM                 ; MOVE DECIMAL
                   FD10       ;   POINT LEFT
        LDFNW      3          ; LOAD LEFT BYTE
                   177400     ;   MASK
        ANDZL      0,3        ; MASK FOR EXPONENT AND SHIFT
        SUBZR      2,2        ; SET BIT 0
        SUB#       2,3,SNR    ; ZERO EXPONENT?
        JMP        OF3        ; YES
        MOVL       3,3,SNC    ; IS THE EXPONENT NEGATIVE?
        JMP        OF3        ; YES
        ISZ        FIELD      ; NO - INCREMENT FIELD SIZE
        JMP        OF2
;    NOW THE VALUE IS < 1.  CONTINUE FORMATTING
OF3     LDA        3,FIELD    ; LOAD INTEGER FIELD SIZE REQUIRED
        DST                   ; STORE THE POSITIVE
                   VAL        ;   NUMBER
        LDA        2,TMP2     ; RELOAD INTEGER F.W.
        SUB        2,3,SZR    ; EXACT FIT?
        JMP        OF4        ; NO
        LDA        2,OSIGN    ; YES - BUT IS THERE A MINUS?
        LDA        3,MSIGN    ; GET ASCII MINUS SIGN
        SUB        2,3,SZR    ; EQUAL?
        JMP        OF6        ; NO - THERE'S ENOUGH ROOM
        JMP        FIT        ; WILL NOT FIT
OF4     MOVL#      3,3,SNC    ; WILL NUMBER FIT?
        JMP        FIT1       ; NO - TRY AND MAKE IT FIT
        INC        3,3        ; ADD PLACE FOR SIGN
        MOV        3,1,SNR    ; BLANKING NECESSARY?
        JMP        OF5        ; NO
        LDA        0,BLANK    ; PAD FIELD WITH LEADING
        JSR        0,PAD      ;   BLANKS
OF5     LDA        0,OSIGN    ; LOAD SIGN CHARACTER
        JSR        0,BYT      ;   AND OUTPUT
```

```
;   OUTPUT THE INTEGEF PART
OF6     DLD                     ; RELOAD THE
                VA              ;   VALUE
        FMPNM                   ; MULTIPLY BY
                FD              ;   10
        DST                     ; SAVE THE MULTIPLIED
                VAL             ;   VALUE
        FIX                     ; GET DIGIT
        MOV     1,2             ; SAVE IT
        FLO                     , FLOAT IT
        FNM
        FSBNM                   ; SUBTRACT OFF THE
                VAL             ;   VALUE
        FNG                     ; MAKE IT POSITIVE
        DST                     ; STORE IT
                VAL
        ADPI    60              ; ADD ASCII " ZERO" TO DIGIT
        MOV     2,0
        JSR     O BYT           ; OUTPUT THE CHARACTER
        DSZ     FIELD           ; CHECK LOOP COUNTER
        JMP     OF6
;   END OF INTEGER OR DECIMAL FORMATTING
        LDA     0,FIRST         ; TEST IF
        MOV     0,0,SZR         ;   DONE
        JMP     OFEX            ; YES - EXIT
,   SET UP TO DO DECIMAL PART
        LDA     3,ORT3          ; GET THE
        LDA     0,1,3           ;   ORIGINAL DECIAML FIELD WIDTH
        MOV     0,0,SNR         ; ZERO?
        JMP     OFEX            ; YES - EXIT
        LDA     0,DCPT          , GET DECIMAL POINT
        JSR     O BYT           ;   AND OUTPUT
        ISZ     FIRST           ; SET FLAG
        LDA     0,NDEC          ; RELOAD DECIMAL FIELD WIDTH
        MOV     0,0,SNR         ; ZERO?
        JMP     OFEX            ; YES - EXIT
        STA     0,FIELD         ; NO  - STORE LOOP COUNTER
        JMP     OF6             ; DO DECIMAL PART
.   NUMBER WILL NOT FIT IN SPECIFIED INTEGER FIELD.
.   TRY AND DECREASE DECIMAL FIELD SIZE AND ADD TO
;   INTEGER FIELD UNTIL NUMBER FITS.
.
FIT1:   NEG     3,3             ; GET 2'S COMPLEMENT
FIT     LDA     0,OSIGN         ; LOAD SIGN TO OUTPUT
        LDA     2,MSIGN         ;    AND MINUS SIGN
        SUB     0,2,SNR         ; EQUAL?
        DEC     3               ; YES - INCREASE NEED
        LDA     1,NDEC          ; LOAD CURRENT DECIMAL FIELD WIDTH
FITL:   DEC     1               ; DECREMENT THE DECIMAL F W.
        COM#    1,1,SNR         ; OUT OF ROOM?
        JMP     ERRS            ; YES - OUTPUT ERROR SIGNS
        INC     3,3,SZR         ; INCREASE INTEGER FIELD
        JMP     FITL            , TRY AGAIN
```

```
;    NUMBER FITS - NOW HAVE NEW INTEGER / DECIMAL FIELDS
         STA      1,NDEC      ; STORE
         MOV      2,2,SNR     ; OUTPUT THE SIGN?
         JSR@      OBYT       ; OUTPUT BYTE
         JMP      OF6
;
ERRS:    LDA      0,DSIGN
         LDA      1,FW
         JSR      O.PAD
OFEX:    LDA      3,ORT3      ; RETURN JUMP
         JMP      2,3
ORT3:    .BLK     1
NDEC:    .BLK     1           ; DECIMAL FIELD WIDTH
FW:      .BLK     1           ; TOTAL FIELD WIDTH
FIRST:   .BLK     1           ; PASS # FLAG
FIELD:   .BLK     1           ; LOOP COUNTER
OSIGN:   .BLK     1           ; SIGN OF OUTPUT NUMBER
VAL  :   .BLK     2           ; NUMBER PASSED TO ROUTINE
DCPT:    56
DSIGN    "$
          RDX     10
FFTAB    0.5
         0.05
         0.005
         0.0005
         0.00005
         0.000005
         0.0000005
         .RDX     8
FDMAX:   7                    ; SIZE OF TABLE

;*************************************************************
;
;    SUBROUTINE O.PAD -  THIS ROUTINE PADS THE OUTPUT BUFFER WITH
;    WITH A CERTAIN NUMBER OF PADDING BYTES.
;
;    AC0 = PADDING BYTE
;    AC1 = LENGTH OF AREA TO BE PADDED
;    JSR   O.PAD
;
O.PAD:   STA      3,ORT5
         MOVL#    1,1,SZC     ; MAKE SURE COUNT IS
         NEG      1,1         ;   POSITIVE
         STA      0,PADB      ; SAVE
         STA      1,PLOP      ; SAVE
         JSR@      OBYT       ; OUTPUT BYTE
         LDA      0,PADB      ; RELOAD PADDING
         DSZ      PLOP        ; SKIP ON ZERO COUNTER
         JMP      .-3         ; AGAIN
         JMP@     ORT5
ORT5:    .BLK     1
PADB:    .BLK     1
PLOP:     BLK     1
```

```
;*************************************************************
;
;    SUBROUTINE O MOV -  THIS ROUTINE TRANSFERS BYTES FROM
;    THE CALLING ROUTINE'S BUFFER INTO THE OUTPUT BUFFER.  A
;    STARTING BUFFER WORD ADDRESS AND BYTE COUNT ARE PASSED AS
;    ARGUMENTS
;
;    AC0 = STRING BYTE STARTING ADDR
;    AC1 = BYTE COUNT
;    JSR  O MOV
;
O MOV    STA      3,ORT6       ; STORE RTA
         STA      1,PLOP       ; STORE COUNT
         STA      0,PADB       ;   AND STORE
MAU      LDA      2,PADB       ; LOAD BYTE POINTER
         MOVZR    2,2          ; GET WORD ADDR
         LDA      0,0,2        ; PICK UP STRING WORD
         MOV      0,0,SZC      ; GET CORRECT BYTE
         MOVS     0,0
         JSR@      OBYT
         ISZ      PADB         ; BUMP BYTE POINTER
         DSZ      PLOP         ; SKIP IF COUNT GOES ZERO
         JMP      MAU          ; ANOTHER ONE
         JMP@     ORT6
ORT6      BLK     1


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;       THIS SUBROUTINE CALCULATES THE SQUARE ROOT OF A
;       NORMALIZED FLOATING POINT ARGUMENT USING NEWTON ITERATION
;
;       ENTRY:  AC0,AC1 CONTAIN ARGUMENT
;
;       EXIT:   AC0,AC1 CONTAIN SQUARE ROOT
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
SQRT:   DST                     ;STORE ARGUMENT
                DST1
        DST                     ;STORE ARGUMENT
                DST2
        STA     3,SQRET         ;SAVE RETURN
        LDA     0,DST1          ;LOAD FIRST WORD OF ARG
        LDA     1,DST1+1        ;LOAD 2ND WORD OF ARG
        ANFNW   0
                377
        STA     0,STO4          ;STORE MOST SIG 8 BITS OF MANTISSA
        MOV     0,0,SZR         ;CHECK FOR ZERO MANTISSA
        JMP     NONZR
        MOV     1,1,SNR
        JMP     ZERR            ;ZERO ARG = ZERO RESULT
```

```
NONZR:  LDA     0,DST1      ;OBTAIN EXPONENT OF ARG
        ANFNW   0                                              ;
                77400                                          ;
        LLSH    1           ;SHIFT LEFT TO PICK UP EXPONENT SIGN
        LDFNW   2                                              ;
                100000                                         ;
        XOR     2           ;GET EXPONENT IN TWOS COMPLEMENT
        MOV     2,0                                            ;
        RASH    7           ;SHIFT EXP TO LOW ORDER BYTE
        RASH    2                                              ;
        MOVZR#  0,0,SZC     ;SKIP IF EXP EVEN, Q = 0
        JMP     Q1          ;JMP IF EXP ODD
        RASH    1           ;HALVE EXP
        LDA     2,.EVEN     ;LOAD POINTER TO EVEN CONSTANTS
        JMP     NEP         ;STORE NEW EXP
Q1:     SUBZL   1,1         ;LOAD 1 INTO AC1
        SUB     1,0         ;SUBTRACT 1 LEAVING 2P
        RASH    1           ;DIVIDE BY 2 GETTING P
        ADD     1,0         ;ADD 1 GETTING P+Q
        LDA     2,.ODD      ;LOAD POINTER TO ODD EXP CONSTANTS
NEP:    STA     0,NEXP      ;STORE P+Q
        LDA     0,ST04                                         ;
        ADFNW   0           ;NEEDED TO MAKE EXCESS 100 WORD
                40000                                          ;
        STA     0,DST2      ;DST2 HAS MANTISSA
        DLDX                ;LOAD C
                4                                              ;
        FADNM               ;COMPUTE C+M
                DST2                                           ;
        DST                                                    ;
                TMP                                            ;
        DLDX                ;LOAD B
                2                                              ;
        FDVNM               ;COMPUTE B/(C+M)
                TMP                                            ;
        DST                                                    ;
                TMP                                            ;
        DLDX                ;LOAD A
                0                                              ;
        FADNM               ;COMPUTE A+B/(C+M)
                TMP                                            ;
        DST                                                    ;
                DST2                                           ;
        LDA     0,DST2      ;LOAD FIRST WORD OF YO
        ANFNW   0           ;MASK OFF ALL BUT EXP
                77400                                          ;
        LLSH    1                                              ;
        LDFNW   2                                              ;
                100000                                         ;
        XOR     2                                              ;
        MOV     2,0                                            ;
        RASH    7                                              ;
        RASH    2
```

```
        LDA     2,NEXP      ;LOAD P+Q
        ADD     2,0         ;ADD YO EXP TO P+Q
        ADFNW   0           ;PUT EXP IN EXCESS 100 FORMAT
                100
        LLSH    10          ;PLACE EXP IN PROPER POSITION
        LDA     1,DST2      ;LOAD 1ST WORD OF YO
        ANFNW   1
                377
        ADD     1,0
        STA     0,DST2
        LDA     2,MN3
        STA     2,STO1
IMPV:   DLD                 ;LOAD X
                DST1
        FDVNM               ;COMPUTE X/Y1
                DST2
        FADNM               ;COMPUTE Y1 + X/Y1
               DST2
        FDVNM               ;COMPUTE (Y1+X/Y1)/2
                F2
        DST                 ;STORE Y2
                DST2
        ISZ     STO1        ;INC COUNTER
        JMP     IMPV        ;IMPROVE ESTIMATE
ZERR    JMP@    SQRET       ;RETURN WITH SQRT IN AC0,AC1
SQRET   0
NEXP:   0
 ODD    AO                  ;POINTER TO ODD COEFFICIENTS
 EVEN:  AE                  ;POINTER TO ODD COEFFICIENTS
MN3:    -3
STO1:   0
STO4:   0
         .RDX     10
DST1:   0.0
DST2:   0.0
T       0.0
AE:     1.80713
        -1.57727
        0.954182
AO:     0.428795
        -0.3430368
        0.877552
         RDX     8
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THIS ROUTINE CALCULATES THE TANGENT OF A FLOATING POINT
;       RADIAN ANGLE
;
;       ENTRY   AC0,AC1 CONTAIN THE ANGLE
;       EXIT    AC0,AC1 CONTAIN TAN
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TAN     STA     3,SQRET         ;SAVE RETURN
        DST                     ;SAVE ANGLE
                DST1
        JSR     COS             ;FIND COS
        DST                     ;SAVE COS
                DST2
        DLD                     ;RELOAD ANGLE
                DST1
        JSR     SIN             ;FIND SIN
        LDA     2,DST2          ; MS 16 BITS OF COS
        MOV#    2,2,SZR         ;SKIP IF COS = 0
        JMP     CNE             ;COS .NE. 0
        MOV     0,2             ; MS 16 BITS OF SIN
        DLD                     ;LARGEST FP NUMBER
                LARGE
        MOVL#   2,2,SZC         ;SKIP IF SIGN(SIN) = 0
        FNG                     ;NEGATE
        JMP@    SQRET           ;RETURN

CNE     FDVNM                   ;FIND TAN
                DST2
        JMP@    SQRET           ;RETURN
LARGE:  077777                  ;LARGEST FP NUMBER
        177777


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THIS ROUTINE CALCULATES EITHER THE SINE OR COSINE OF A
;       FLOATING POINT RADIAN ANGLE.
;
;       ENTRY:  AC0,AC1 CONTAIN THE ANGLE
;
;       EXIT:   AC0,AC1 CONTAIN THE SIN/COS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
COS:    SUBZL   2,2             ;AC2 = 1 FOR COS, 0 FOR SIN
        JMP     SNCO
SIN:    SUBOL   2,2
SNCO:   DST                     ;STORE ARGUMENT
                DSTR1
        STA     3,CSRET         ;SAVE RETURN
        ANFNW   0               ;PUT SIGN BIT IN AC0
                100000
        RLSH    17              ;SHIFT SIGN BIT
        MOVZL   2,2,SZR         ;AC2 HAS 2Q
        SUB     0,0
        STA     0,STOR1         ;STORE SIGN OF ARG
        SUB     0,0             ;AC0 = 0
```

```
        STA     0,STOR2     ;STORE ZERO
        LDA     0,DSTR1     ;LOAD FIRST WORD OF ARG
        ANFNW   0           ;REMOVE SIGN BIT
                077777      ;TAKE ABS OF ARG
        STA     0,DSTR1     ;DSTR1 = ABS(ARG)
        STA     2,STOR3     ;STORE 2Q
        DLD                                                 ;
                DSTR1                                       ;
        FMPNM               ;ABS(ARG)*4/PI
                PI 4                                        ;
        DST                 ;STORE VALUE IN DSTR2
                DSTR2                                       ;
        FIX                                                 ;
        DST                 ;DP INTEGER IN DSTR3
                -DSTR3                                      ;
        FLONM               ;FP INTEGER IN DSTR1
        DST                                                 ;
                DSTR1       ;FP INTEGER IN DSTR1
        DLD                                                 ;
                DSTR2                                       ;
        FSBNM               ;REMOVE INTEGER PORTION
                DSTR1                                       ;
        DST                 ;FP FRACTION IN DSTR4
                DSTR4                                       ;
        LDA     0,DSTR3+1   ;LOAD I
        LDA     1,STOR3     ;LOAD 2Q
        ADDZR   0,1         ;(I+2Q)/2
        STA     1,STOR3                                     ;
        MOVR    0,0,SNC     ;CHECK FOR ODD OR EVEN I
        JMP     EVEN                                        ;
        DLD                                                 ;
                F1                                          ;
        FSBNM               ;(1-G)
                DSTR4                                       ;
        DST                 ;STORE -(G-1)
                DSTR4                                       ;
        LDA     1,STOR3     ;LOAD (I+2Q)/2
        INC     1,1         ;ADD 1
EVEN:   MOVR    1,1,SZC     ;DETERMINE SIN/COS POLYNOMAL CONSTANT
        JMP     +2                                          ;
        DSZ     STOR2       ;SET TO -1 FOR SIN
        DLD                 ;COMPUTE R*R
                DSTR4                                       ;
        FMPNM                                               ;
                DSTR4                                       ;
        DST                 ;STORE R*R
                DSTR2                                       ;
        LDA     2, SNC      ;POINTER TO SIN CONSTANTS
        ISZ     STOR2                                       ;
        LDA     2,.CSC      ;POINTER TO COS CONSTANTS
        LDFNW   1           ;INITIALIZE DEGREE OF POLYNOMIAL
                3                                           ;
        STA     1,STOR4                                     ;
        DLDX                ;LOAD 1ST COEF
                0                                           ;
        DST                 ;STORE 1ST CODE
                DSTR3
```

```
PLO     FMPNM                   ;COMPUTE AR
                DSTR2
        DST
                DSTR5
        ADPI    2               ;ADD IMMEDIATE TO AC2
        DLDX                    ;LOAD NEXT COEF
                0
        FADNM
                DSTR5
        DST
                DSTR3
        DSZ     STOR4           ;DECREMENT COUNTER
        JMP     PLO             ;CONTINUE POLY EVALUATION
        LDA     2,STOR2         ;CHECK FOR SIN OR COS
        MOV     2,2,SZR         ;IF SIN, COMPUTE P(R*R)*R
        JMP     CONT            ;JMP COS
        DLD
                DSTR3
        FMPNM
                DSTR4
        DST
                DSTR3
CONT:   LDA     1,STOR3         ;LOAD (I+2Q)/2
        MOVZR   1,1             ;(I+2Q)/4
        LDA     0,STOR1         ;LOAD SIGN
        MOVR    0,0,SNC
        INC     1,1
        MOVR    1,1,SZC
        JMP     OTPU
        LDA     0,DSTR3
        ADFNW   0
                100000
        STA     0,DSTR3
OTPU:   DLD
                DSTR3
        ADDOL# 0,0,SNR          ;CHECK FOR FLOATING POINT NEG. ZERO
        SUB 0,0                 ;FIX IT
        JMP@    CSRET           ;RETURN WITH SIN/COS IN AC0,AC1
CSRET:  0
.SNC:   SINC
.CSC:   CSC
STOR1:  0
STOR2:  0
STOR3:  0
STOR4:  0
        .RDX    10
DSTR1:  0.0
DSTR2:  0.0
DSTR3:  0.0
DSTR4:  0.0
DSTR5:  0.0
PI.4:   1.2732395               ;4/PI
```

```
SINC    -0.35950439E-4
        0.2490001007E-2
        -0.807454325E-1
        0.78539816
C       -0.31872783E-3
        0.1584968416E-1
        -0.3084241655
        0.99999996
         RDX     8

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       THIS ROUTINE FINDS THE ABSOLUTE VALUE OF A FLOATING
;         POINT ARGUMENT.
;
;       ENTRY:   ACO,AC1 CONTAIN THE ARGUMENT
;       EXIT:    ACO,AC1 CONTAIN THE ABSOLUTE VALUE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ABS     ANFNW   0           ;REMOVE SIGN BIT
                077777
        JMP     0,3         ;RETURN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       A R C T A N   C A L C U L A T I O N
;
;       2 ENTRY POINTS: ATAN,ATAN2
;
;       ATAN
;       ----
;           RANGE: (-PIE/2,+PIE/2)
;
;       ENTRY:   ACO,AC1 CONTAIN INPUT PARAMETER
;
;       EXIT:    ACO,AC1 CONTAIN THE ARCTAN
;
;       ATAN2
;       ----
;            RANGE: (0,2*PIE)
;
;       CALLING SEQUENCE:    JSR    ATAN2
;                            XADD         ;ADDRESS OF X
;                            YADD         ;ADDRESS OF Y
;                            <NORMAL RETURN>
;
;
;       EXIT:    ACO,AC1 CONTAIN ARCTAN(X/Y)
```

```
;
;       FOR ABS(X) > 1, THE FORMULA
;       ATAN(ABS(X)) = PIE/2 - ATAN(1/ABS(X)) IS USED.
;
;       FOR ABS(X) < 1, THE FORMULA
;       ATAN(X)=PIE/8+ATAN((X-TAN(PIE/8))/(1+X*TAN(PIE/8)))
;       IS USED FOR RANGE REDUCTION TO (0,TAN(PIE/8)).
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ATRET   0                       ;RETURN ADDRESS
ATIND   0                       ;0 FOR ATAN, 1 FOR ATAN2
ATAN2:  SUBZL   2,2             ;1 FOR ATAN2
        STA     2,ATIND
        LDA     2,0,3           ;GET ADDRESS OF X
        DLDX                    ;LOAD X
                0
        DST                     ;SAVE X
                ATX
        LDA     2,1,3           ;GET ADDRESS OF Y
        DLDX                    ;LOAD Y
                0
        DST                     ;SAVE Y
                ATY
        INC     3,3
        INC     3,3
        STA     3,ATRET         ;SAVE RETURN
        MOV#    0,0,SNR         ;SKIP IF Y .NE. 0
        JMP     TOVF            ;OVERFLOW
        DLD
                ATX
        FDVNM                   ;GET X/Y
                ATY
        JMP     ARCT            ;BRANCH TO REGULAR ARCTAN CODE

TOVF:   DLD                     ;LOAD PIE/2
                PIE2
        LDA     2,ATX           ; MS 16 BITS OF X
        MOVL#   2,2,SNC         ;SKIP IF X WAS -VE
        JMP@    ATRET           ;RETURN
        FADNM                   ;GET 3/2*PIE
                PIE
TRET:   JMP@    ATRET           ;RETURN

ATAN:   SUBOL   2,2             ;0 FOR ATAN
        STA     2,ATIND
        STA     3,ATRET         ;SAVE RETURN
ARCT:   DST                     ;SAVE ARGUMENT
                AT2
        ANFNW   0               ;FIND ABSOLUTE VALUE
                077777
        DST                     ;SAVE ABS(X)
                AT1
        FSBNM                   ;SUBTRACT FP1
                F1
        MOVL#   0,0,SZC         ;SKIP IF .GE. 0
        JMP     LTONE           ;ABS(X) .LT. 1
        DLD                     ;LOAD FP1
                F1
```

```
        FDVNM               ;FORM 1/ABS(X)
                AT1
        DST                 ;SAVE NEW ARG
                AT1
LTONE   SUB     0,0         ;ZERO P8
        STA     0,P8
        DLD                 ;RELOAD ARGUMENT
                AT1
        FSBNM               ;FORM X - TAN(PIE/8)
                TPIE8
        MOVL#   0,0,SZC     ;SKIP IF X .GE. TAN(PIE/8)
        JMP     LTP8
        DLD                 ;RELOAD ARG
                AT1
        FMPNM               ;TIMES TAN(PIE/8)
                TPIE8
        FADNM               ;ADD FP 1
                F1
        DST                 ;SAVE
                AT3
        DLD                 ;RELOAD ARG
                AT1
        FSBNM               ;SUBTRACT TAN(PIE/8)
                TPIE8       ;SUBTRACT TAN(PIE/8)
        FDVNM               ;DIVIDE BY 1+X*TAN(PIE/8)
                AT3
        DST                 ;SAVE NEW ARG
                AT1
        SUBZL   0,0         ;+1
        STA     0,P8        ;INDICATE ABS(X) .GE. TAN(PIE/8)
LTP8:   DLD                 ;LOAD ARG
                AT1
        FMPNM               ;SQUARE ARGUMENT
                AT1
        LDA     2, ATC      ;POINTER TO CO-EFFICIENTS
        JSR     POLY        ;CALCULATE POLYNOMIAL
        FMPNM               ;FORM X*P(X**2)
                AT1
        DST                 ;SAVE RESULT
                AT1
        LDA     0,P8
        MOV     0,0,SNR     ;SKIP IF X .GT. PIE/8
        JMP     OK
        DLD                 ;LOAD RESULT
                AT1
        FADNM               ;ADD PIE/8
                PIE8
        DST                 ;NEW RESULT
                AT1
OK:     DLD
                AT2
        ANFNW   0           ;FIND ABS VALUE
                077777
```

```
        FSBNM                   ;FORM ABS(X) - 1
                F1
        MOVL#   0,0,SZC         ;SKIP IF ABS(X) .GE. 1
        JMP     ALT1            ;ABS(X) .LT. 1
        DLD
                PIE2
        FSBNM                   ;FORM PIE/2 - ATAN(1/ABS(X))
                AT1
        DST                     ;SAVE RESULT
                AT1
ALT1:   DLD                     ;LOAD RESULT
                AT1
        LDA     2,ATIND         ;GET ATAN/ATAN2 INDICATOR
        MOV#    2,2,SNR         ;SKIP IF ATAN2
        JMP     ALT2            ;ATAN
        LDA     2,ATX           ;GET SIGN OF X
        LDA     3,ATY           ;GET SIGN OF Y
        MOVL#   2,2,SNC         ;SKIP IF X -VE
        JMP     XPLUS           ;X +VE
        MOVL#   3,3,SNC         ;SKIP IF Y -VE
        JMP     A2NEG
ADPYE:  FADNM                   ;ADD PIE
                PIE
        JMP@    ATRET           ;RETURN
A2NEG:  FNG                     ;NEGATE
        MOVL#   2,2,SNC         ;SKIP IF X -VE
        JMP     ADPYE           ;ADD PIE
        FADNM                   ;ADD 2*PIE
                TPIE
        JMP     TRET            ;RETURN
XPLUS:  MOVL#   3,3,SZC         ;SKIP IF Y +VE
        JMP     A2NEG           ;RESULT -VE
        JMP     TRET            ;RETURN
ALT2:   LDA     2,AT2
        MOVL    2,2,SZC         ;SKIP IF +VE
        FNG                     ;NEGATE
        JMP     TRET            ;RETURN
ATX:    0 0                     ;ATAN2 X
ATY:    0 0                     ;ATAN2 Y
 ATC:   ATC
ATC:    4
        0.79866237E-1
        -0.13852054
        0.19974467
        -0.33332799
        0.999999982
AT1:    0.0
AT2:    0.0
AT3:    0.0
PIE8:   0.39269908
TPIE8:  0.41421356
P8:     0                       ;ZERO IF X < PIE/8
```

```
************************************************************************
*                                                                      *
*    ****  **    *  *  ****  **                                        *
*    *    *  *   *  *  *     *  *                                      *
*    * **  *  *  ****  ***   *  *       PROGRAM                        *
*    *  *  *  *  *  *  *     *  *                                      *
*    ****  **    *  *  ****  **                                        *
*                                                                      *
************************************************************************
```

THE GDHED ROUTINE DOES THE MATHEMATICAL CALCULATIONS OF THE GDHED SYSTEM AND OUTPUTS THE RESULTS TO THE AUXILIARY EQUIPMENT

```
         TITL FAME1
         NREL
         ENT START
         ENT LOOP
         ENT ACR1
         EXTN  PTY  EXTN FMUL  DEN
         EXTN ATAN2 SQRT
         EXTD  OSET  OFLT
         EXTN SEC COUNT  QUIT  FORK
         EXTN  IOX  QUIT

START     FORK
         10
         COUNT
          PTY
         100
LOOP      IOX                  ;DOPPLER I/O
         23
         001300
         ACR1
         10
         ER1
          IOX
         
         001300
         ACR2
         315
         ERROR
         LDFNW 0
         0
         STTNA 0
         MARK 4
         STTNA 0
         J04
ACR1     LDFMA 2
         MARK 4
         LDFNW 1
         14
```

```
          JMP @ +2
          JMP   +2
          11
AH1       DLDX
          ADR1
          LDFNA 2
          J04
          LDFNX 3
          CMASS
          LDFNW 2
          377
          AND 2,0
          SUB 3,0,SZR
          JMP ABORT
          LDFNW 2
          077770
          AND 1,2
          STTNA 2
          SHDG
          LDFNW 2
          060000
          AND 1,2
          STTNA 2
          HOLE1
          LDFNW 1
          0
          ADD 1,2,SZR
          JMP A
          LDFNA 0
          SHDG
          RLSH 3
          MOV 0,1
          JMP SCAL
A:        LDFNA 2
          HOLE1
          LDFNW 1
          060000
          SUB 1,2,SZR
          JMP ABORT
          LDFNA 0
          SHDG
          LLSH 3
          RLSH 6
          LDFNW 2
          176000
          ADD 2,0
          MOV 0,1
          JMP SCAL1
HOLE1 0
SCAL:     LDFNW 0
          0
          FLO
          JMP NEXT4
ABORT:  JMP @.LOOP
 LOOP:LOOP
 AHO:AHO
MARK4:0
J04:0
SHDG:0
```

```
 ABORT  ABORT
SCAL1   LDFNW 0
        177777
        FLO
NEXT4   LDFNA 2
        MARK4
        DSTX
        DSP
        INC 2,2
        INC 2,2
        STTNA 2
        MARK4
        LDFNA 2
        J04
        INC 2,2
        STTNA 2
        J04
        JMP @ AHO
ADR1:   BLK 30
CMAS3: 77
277
137
337
037
237
117
317
57
257
157
357
I1:     LDFNA 2
        MARK4
        LDFNW 1
        30
        SUB 2,1,SZR
        JMP VEL2
        JMP @ I2
VEL2:   DLDX
        ADR1
        STTNA 0
        TEMP1
        LDFNW 2
        377
        AND 2,0
        LDFNA 2
        J04
        LDFNX 3
        CMAS3
        SUB 0,3,SZR
        JMP ABORT
        LDFNW 2
        140000
        LDFNA 0
```

```
         TEMP1
         AND 2,0
         RLSH 14
         STTNA 0
         MINVS
         LDFNW 2
         077777
         AND 1,2
         STTNA 2
         HOLE
         LDFNW 2
         060000
         AND 1,2
         STTNA 2
         HOLE1
         LDFNW 1
         0
         ADD 1,2,SZR
         JMP C
         LDFNA 0
         HOLE
         LLSH 2
         LDFNA 1
         MINVS
         ADD 0,1
         JMP SCAL5
TEMP1:0
HOLE:0
MINVS:0
C:       LDFNW 1
         060000
         SUB 1,2,SZR
         JMP @ ABORT
         LDFNA 0
         HOLE
         LLSH 2
         LDFNA 1
         MINVS
         ADD 0,1
         JMP SCAL6
 I2:I2
SCAL5:   LDFNW 0
         0
         FLO
         FDV
         TEN
         JMP NEXT3
SCAL6:   LDFNW 0
         177777
         FLO
         FDV
         TEN
NEXT3:   LDFNA 2
         MARK4
         DSTX
         DSP
```

```
          INC 2,2
          INC 2,2
          STTNA 2
          MARK4
          LDFNA 2
          I04
          INC 2,2
          STTNA 2
          I04
          JMP @ I1
TEN  10.
 I1  I2
DSP    BLK 30
I2        LDFNA 2
          MARK8
          DLDX
          DSP
          DST
          SP
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          CP
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          SR
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          CR
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          SH
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          CH
          INC 2,2
          INC 2,2
          DLDX
          DSP
          DST
          VX
```

```
INC 2,2
INC 2,2
DLDX
DSP
DST
VY
INC 2,2
INC 2,2
DLDX
DSP
DST
VZ
INC 2,2
INC 2,2
DLDX
DSP
DST
VH
INC 2,2
INC 2,2
DLDX
DSP
DST
VD
INC 2,2
INC 2,2
DLDX
DSP
DST
VV
LDFNW 0
0
STTNA 0
MARK4
LDFNA 2
MARK4
DLDX
ADR2
LDFNA 3
CMAS2
SUB 3,0,SZR
JMP @ NG
LDFNW 0
9
ADD 0,2
DLDX
ADR2
FLD
FDV
DENO1
FMP
NUM1
FST
GLATT
```

```
        INC
        INC
        DEP
        ADR2
        FLO
        FDV
        DENO1
        FMP
        NUM1
        FST
        DEUNO
        INC 2,2
        INC
        LDENW
        A2
        ADD
        LDENX 1
        ADR2
        LDENA 0
        MASK1
        AND 1,0,S7R
        JMP NEG2
        FLO
        FDV
        DENO2
        FMP
        NUM2
        FST
        GVN
        INC 2,2
        LDENX 1
        ADR2
        LDENC 0
        MASK1
        AND 1,0,S7R
        JMP NEG1
DF1     FLO
        FDV
        DENO2
        FMP
        NUM2
        FST
        GVE
        INC 2,2
        INC 2,2
        INC 2,2
        LDENX 1
        ADR2
        LDENA 0
```

```
        GSTAT
        JMP CALC
 NG NG
DEG2    LDFNW 0
        177777
        JMP OK
DEG1    LDFNW 0
        177777
        JMP OK1
CALC    FLD
        VX
        FMP
        VX
        FST
        LOT1
        FLD
        VY
        FMP
        VY
        FAD
        LOT1
        FST
        LOT1
        FLD
        VZ
        FMP
        VZ
        FAD
        LOT1
        FST
        LOT1
        FLD
        GVN
        FMP
        VD
        FST
        ST1
        FLD
        VH
        FMP
        GVE
        FSB
        ST1
        FST
        ST2
        FLD
        VD
        FMP
        GVE
        FST
        ST1
        FLD
        GVN
        FMP
        VH
        FAD
```

```
ST1
FST
ST1
 ENQU
FMLOK
LDFNA 0
DAVE
JSR @ OSET
FLD
SEC
JSR @ OFLT
[illegible]
[illegible]
 DEQU
FMLOK
 IOX
0
140000
NOTE1*2
14
ERROR
 ENQU
FMLOK
LDFNA 0
DAVE7
JSR @ OSET
JSR @ OTAN
SH
CH
FST
ST4
JSR @ OFLT
2
2
 DEQU
FMLOK
 IOX
0
140000
NOTE7*2
12
ERROR
 ENQU
FMLOK
LDFNA 0
DAVE6
JSR @ OSET
JSR @ OTAN
ST2
ST1
FST
ST3
JSR @ OFLT
2
2
```

```
 DEQU
FMLOK
 IOX
0
140000
NOTE6*2
12
ERROR
FLD
ST3
FSB
ST4
FST
LOT2
 ENQU
FMLOK
LDFNA 0
DAVE3
JSR @.OSET
FLD
LOT2
JSR @.OFLT
3
2
 DEQU
FMLOK
 IOX
0
140000
NOTE3*2
14
ERROR
 ENQU
FMLOK
LDFNA 0
DAVE1
JSR @.OSET
FLD
LOT1
JSR @.SQRT
JSR @.OFLT
5
1
 DEQU
FMLOK
 IOX
0
140000
NOTE2*2
11
ERROR
 ENQU
FMLOK
LDFNA 0
DAN2
```

```
          JSR @ OSET
          FLD
          GSTAT
          JSR @ OFLT
          2
          1
           DEOU
          FMLOK
           IOX
          0
          140000
          MAN2*2
          .
          ERROR
NG        LDFNW 0
          0
          STTNA 0
          MARK8
          JMP @ +2
          JMP  +2
          LOOP
ERROR   QUIT
 OTAN ATAN2
 SQRT SQRT
DAVE NOTE1*2
NOTE1    TXT "            <40><40>"
DAVE7 NOTE7*2
DAN2 MAN2*2
MAN2    TXT "      <15><12>"
DENO1 2147483747 0
NUM1 180.0
GLATT   .BLK 2
CMAS2 147155
GLONG   .BLK 2
MASK2: 077777
MASK1 100000
NUM2 159.995117
DENO2 32767 0
GVN:   BLK 2
GVE.   BLK 2
GSTAT   .BLK 2
ADR2:   BLK 316
NOTE7   .TXT "         <40><40>"
DAVE1 NOTE2*2
NOTE2:   TXT "         <40><40>"
DAVE6 NOTE6*2
NOTE6    TXT "         <40><40>"
DAVE3 NOTE3*2
NOTE3:   TXT "            <40><40>"
ST1:   BLK 2
ST2:   BLK 2
ST3:  .BLK 2
ST4:   BLK 2
LOT2:    BLK 2
LOT1     BLK 2
```

```
MARK 3 0
ER1 HALT
SP   BLK 2
CP    BLK 2
SR    BLK 2
CR    BLK 2
SH    BLK 2
CH    BLK 2
VX    BLK 2
VY    BLK 2
VZ    BLK 2
VH    BLK 2
VD    BLK 2
VV    BLK 2
 END

          .TITL TOCK
           NREL
           EXTN  WAIT
           EXTN .PTY
          .ENT SEC COUNT

COUNT     .WAIT
          1
          FLD
          SEC
          FAD
          B
          FST
          SEC
          JMP COUNT
SEC: 0 0
B: 0. 1
 END
```